VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Rozpoznávání lidských činností pomocí Deep Learning

Human Activity Recognition Using Deep Learning

2019

Radek Kopecký

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Radek Kopecký**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Rozpoznávání lidských činností pomocí Deep Learning
Human Activity Recognition Using Deep Learning

Jazyk vypracování: čeština

Zásady pro vypracování:

Rozpoznávání akcí a činností je v poslední době věnována značná pozornost výzkumu. Úspěšné metody mohou být využity v mnoha aplikacích, jako je bezpečnost, analýza lidského chování apod. Je proto potřeba vyvíjet spolehlivé metody pracující v reálném čase.
Vedle rozpoznávání akcí se výzkum v deep learningu rozšířil do spoustu oborů a dosahuje zajímavých detekčních výsledků, přičemž zpracování lze provádět i na GPU a tím ušetřit čas.

1. Seznamte se s metodami používanými v Deep learningu jako jsou konvoluční neuronové sítě a náležitě je popište.
2. Seznamte se s metodami detekce a klasifikace akcí.
3. Sestavte a odzkoušejte konvoluční neuronovou síť s využitím zvoleného frameworku na rozpoznávání lidských činností.
4. Zjištěné poznatky řádně zdokumentujte v textu práce.

Seznam doporučené odborné literatury:

[1] Chapter 10. Neural Networks. The Nature of Code [online]. Dostupné z: http://natureofcode.com/book/chapter-10-neural-networks/
[2] JUERGEN, Schmidhuber. Deep Learning in Neural Networks: An Overview. ArXiv [online]. 2015, 88 DOI: 10.1016/j.neunet.2014.09.003. Dostupné z: http://arxiv.org/abs/1404.7828
[3] Convolutional Neural Networks (LeNet). Deep Learning: . . . moving beyond shallow machine learning since 2006! [online]. 2016 Dostupné z: http://deeplearning.net/tutorial/lenet.html
[4] SZEGEDY, Christian, WEI LIU, YANGQING JIA, et al. Going deeper with convolutions. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) [online]. IEEE, 2015, s. 1-9 [cit. 2017-10-12]. DOI: 10.1109/CVPR.2015.7298594. ISBN 978-1-4673-6964-0. Dostupné z: http://ieeexplore.ieee.org/document/7298594/
[5] LECUN, Yann, Léon BOTTOU, Yoshua BENGIO a Patrick HAFFNER. Gradient-Based Learning Applied to Document Recognition [online]. 1998, 46 Dostupné z: http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf
[6] YANG, Rui a Ruoyu YANG. DMM-Pyramid Based Deep Architectures for Action Recognition with Depth Cameras [online]. Dostupné z: http://vigir.missouri.edu/~gdesouza/Research/Conference_CDs/ACCV_2014/pages/PDF/358.pdf

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Radek Simkanič, DiS**

Datum zadání: 01.09.2018

Datum odevzdání: 30.04.2019

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 30. dubna 2019 ..... Kopecký .....

Chtěl bych poděkovat mému vedoucímu bakalářské práce Ing. Radkovi Simkaničkovi, DIS za cenné rady a odborné vedení při zpracovávání této práce. Dále bych chtěl poděkovat svým přátelům a rodině za jejich psychickou podporu v průběhu studia.

Abstrakt

Tématem této bakalářské práce je rozpoznávání lidských činností pomocí Deep learning. V rámci práce jsou popisovány principy neuronových sítí a metody pro detekci a klasifikaci akcí. Následuje implementace, v jejímž rámci byly sestaveny a odzkoušeny dvě konvoluční neuronové sítě podle architektur LeNet a GoogLeNet. Pro odzkoušení sítí byly využity datasety lidských činností MSR a UTKinect. Implementace sítí byla provedena v jazyce Python s využitím frameworku Tensorflow.

Klíčová slova: Deep learning, rozpoznávání lidských činností, neuronové sítě, CNN, Python, Tensorflow

Abstract

The theme of this bachelor's thesis is human activity recognition using Deep Learning. The thesis describes principles of neural networks and methods for action detection and classification. This is followed by implementation, in which two convolutional neural networks based on LeNet and GoogLeNet architectures are created and tested. For testing, human action datasets MSR and UTKinect were used. Implementation was done using programming language Python in combination with the framework Tensorflow.

Key Words: Deep learning, human activity recognition, neural networks, CNN, Python, Tensorflow

Obsah

Seznam použitých zkratek a symbolů

DL	–	Deep Learning - Hluboké učení
AI	–	Artificial Intelligence - Umělá inteligence
GPU	–	Graphic Processing Unit - Grafický procesor (grafická karta)
TPU	–	Tensor Processing Unit - Tensorový výpočetní procesor - procesor speciálně určený pro AI aplikace
GPGPU	–	General-purpose computing on graphics processing units - Způsob využití grafických procesorů pro obecné výpočty a aplikace
OpenCV	–	Open source computer vision - knihovna pro počítačové vidění
CUDA	–	Compute Unified Device Architecture - paralelní výpočetní platforma umožňující stavět aplikace na GPU výpočtech
MP Neuron	–	McCulloch-Pitts model neuronu
MLP	–	Multi-Layer Perceptron - Vícevrstvá neuronová síť
RNN	–	Recurrent Neural Network - Rekurentní neuronová síť
CNN	–	Convolutional Neural Network - Konvoluční neuronová síť
LTU	–	Linear Threshold Unit - Lineární prahová jednotka
ReLU	–	Rectified Linear Unit - Rektifikovaná lineární jednotka
BP	–	Backpropagation - Algoritmus zpětného šíření
STIP	–	Space-Time Interest Points - Časoprostorové zájmové body
HOG	–	Histogram of Oriented Gradients - Histogram orientovaných gradientů
SSM	–	Self Similarity Matrix - Matice vlastních podobností
ROI	–	Region of Interest - Region zájmů
LRCN	–	Long-Term Recurrent Convolutional Network - Dlouhodobě rekurentní konvoluční sítě
TCN	–	Temporal Convolutional Network - Temporální konvoluční sítě
LSTM	–	Long short-term memory - Dlouze krátkodobá paměťová jednotka
LOOCV	–	Leave one sequence out cross validation - Validace vynecháním jednoho vzorku
ILSVRC	–	ImageNet Large Scale Visual Recognition Competition - Soutež pro rozpoznávání obrázků
MSR	–	MSR Action3D Dataset - Dataset lidských akcí
UTKinect	–	UTKinect-Action3D Dataset - Dataset lidských akcí
CIFAR-10	–	Canadian Institute For Advanced Research dataset - dataset obrázků
GNARL	–	GeNeralized Acquisition of Recurrent Links - evoluční algoritmus

hyperNEAT	–	Hypercube-based NeuroEvolution of Augmented Topologies - evoluční algoritmus
K-NN	–	K-Nearest Neighbor - Ktý nejbližší soused (algoritmus strojového učení)
SVM	–	Support Vector Machine - Metoda podpůrných vektorů (algoritmus strojového učení)
CNTK	–	Microsoft Cognitive Toolkit - Knihovna pro DL metody
F_1	–	Harmonický průměr
TP	–	True Positive - Skutečně pozitivní hodnota
FP	–	False Positive - Falešně pozitivní hodnota
FN	–	False Negative - Falešně negativní hodnota

Seznam obrázků

Seznam tabulek

1 Úvod

Rozpoznávání akcí a činností jsou důležité disciplíny počítačového vidění, kterým je zejména v poslední době věnována značná pozornost výzkumu. Úspěšné metody pro rozpoznávání akcí lze využít v širokém spektru aplikací, jako například bezpečnost nebo analýza lidského chování. Problematika rozpoznávání akcí může být rozdělena zejména na detekci a následnou klasifikaci akcí. Obě tyto části skrývají své vlastní typy problémů a existuje široká škála metod pro jejich řešení. V tomto ohledu jsou v poslední době populární zejména metody Deep learningu.

Deep learning je obor v rámci strojového učení a umělé inteligence založený na mnoha různých algoritmech, jehož snahou je přiblížit se způsobu lidského myšlení a umožnit tak počítačům plnit úlohy, které donedávna dokázali efektivně vykonávat pouze lidé. V porovnání s běžnými metodami strojového učení dokáže Deep learning pracovat s předem nezpracovanými daty a učit se rozpoznávat vzory v těchto datech.

Cílem této bakalářské práce je popis současných možností pří rozpoznávání lidských akcí a jejich zpracování pomocí metod Deep learningu. Práce bude rozdělena do několika částí. V první části budou popsány principy neuronových sítí včetně některých typů těchto sítí. V druhé části pak budou podrobněji popsány principy konvolučních neuronových sítí. Třetí část bude věnována popisu některých významných architektur konvolučních neuronových sítí. V čtvrté části bude následovat popis principů metod pro detekci a klasifikaci akcí. Některé metody detekce a klasifikace akcí budou v této části blíže přiblíženy. Poslední část práce pak bude zaměřena na vlastní implementaci algoritmu pro rozpoznávání lidských činností. V rámci této části budou testovány dva vlastní modely konvolučních neuronových sítí. Modely budou testovány z hlediska úspěšnosti při rozpoznávání akcí a pro tyto účely budou využity dva datasety lidských akcí. Konec této části bude zaměřen na vyhodnocení a porovnání výsledků obou sítí.

2 Neuronové sítě

Neuronové sítě jsou propojené výpočetní systémy, jejichž princip vychází ze způsobu zpracování informací v mozku. Historie neuronových sítí sahá až do roku 1943, kdy neurofiziolog Warren S. McCulloch a logik Walter Pitts vyvinuli první konceptuální model umělé neuronové sítě. V něm popsali koncept neuronu - jedné buňky uspořádatelné do sítě, jejíž funkcí je přijímat vstupy, zpracovávat je a generovat výstup. Cílem jejich práce nebyl přesný popis biologické funkčnosti mozku, ale spíše návrh výpočetního modelu, který by sloužil k řešení určitých typů problémů. [1]

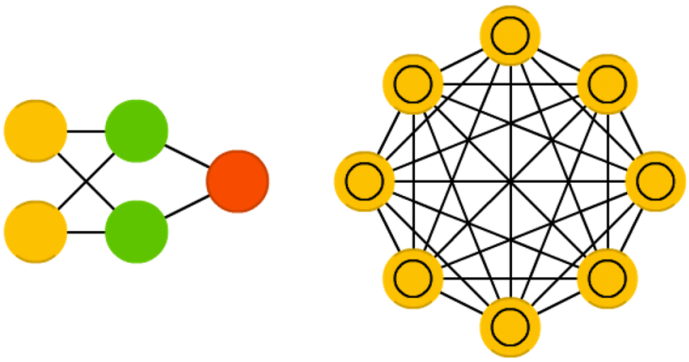
Existují problémy, které jsou pro počítače, v porovnání s lidmi, velmi snadné na řešení, jako například matematické výpočty. Na druhou stranu však existují i problémy, které běžný člověk zvládá řešit velice snadno, ale počítače je řeší jen obtížně. Mezi tyto problémy patří například rozpoznávání obličejů, znaků nebo obecně vzorců. Pro tyto problémy se v dnešní době využívají právě neuronové sítě. [1]

V současné době existuje mnoho rozdílných typů neuronových sítí. Jednotlivé typy sítí se mezi sebou liší například ve struktuře základních funkčních principech nebo zvolené metodě pro učení sítě. Mezi tyto typy se řadí například perceptron, vícevrstvé sítě, konvoluční sítě a nebo rekurentní neuronové sítě.

2.1 Neuron

Základní jednotkou v neuronové síti je neuron, často označovaný také jako jednotka nebo uzel. Neurony jsou propojovány pomocí ohodnocených hran, takto propojené neurony tvoří síť. Váhy těcho hran reprezentují sílu spojení mezi jednotlivými neurony. [2]

Neurony v neuronových sítích bývají typicky organizovány do vzájemně provázaných vrstev. Vrstva je definována jako skupina neuronů, které jsou propojeny jen s neurony z ostatních skupin, nikoliv však ze své vlastní. Vrstvy mohou být rozlišovány podle jejich pozice a propojení na vstupní, skryté a výstupní. Vstupní a výstupní vrstvy jsou první a poslední vrstva v síti. Tyto vrstvy zprostředkovávají datové vstupy a výstupy sítě a mohou být sestaveny odlišným způsobem než skryté vrstvy. Skryté vrstvy jsou pak všechny vrstvy mezi vstupní a výstupní vrstvou. Ne všechny sítě však musí být organizovány do vrstev. Příkladem mohou být třeba Hopfieldovy sítě (Zobrazena na obrázku 1b). [2, 3]



(a) Feed-Forward síť (b) Hopfieldova síť

Obrázek 1: Ukázka organizace neuronů v sítích. [4]

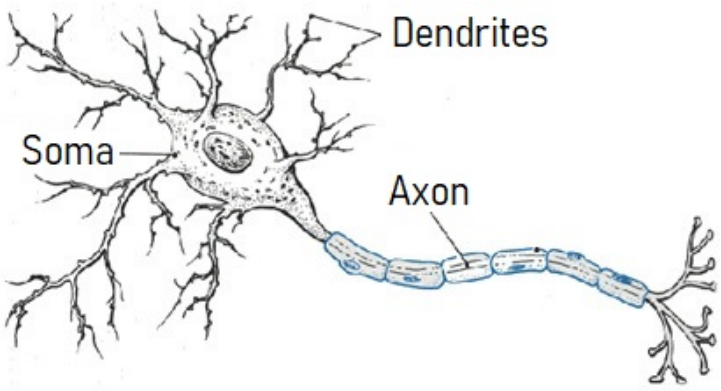
2.2 Biologický neuron

Neurony v živých organizmech jsou buňky, které dokáží pracovat se signály a reagovat na ně. Existuje mnoho typů neuronových buněk, avšak všechny mají společné jisté části - dendrity, somu, axon a synapse. Tyto části jsou popsány na obrázku 2. [5]

Dendrity jsou tenké výstupky ze somy, které slouží jako vstupní kanály pro signály z okolí. Každý neuron má vícero dendritů. Soma je tělo neuronu, ve kterém dochází ke zpracování signálů. Axon je delší výstupek, který slouží pro vysílání signálu. Konec axonu je rozvětvený. Synapse jsou pak propojení mezi dendrity a axonem jednotlivých neuronů. [5]

Funkce neuronu probíhá v následujících krocích. Dendrity neuronu zachycují signály, které se dále šíří dovnitř buňky. V těle buňky dochází k součtu příchozích signálů a vzniká potenciál. Pokud je tento potenciál dostatečně velký, neuron vyšle nervový impuls po axonu. [6]

Lidská nervová soustava obsahuje přibližně 10^{11} - 10^{12} neuronů. Toto číslo se však s rostoucím věkem postupně snižuje. Každý neuron může mít řádově i stovky tisíc dendritů. Pokud část neuronů odumře, lze tuto ztrátu částečně nahradit zvýšením počtu synapsí mezi neurony. [7]



Obrázek 2: Schéma biologického neuronu [7]

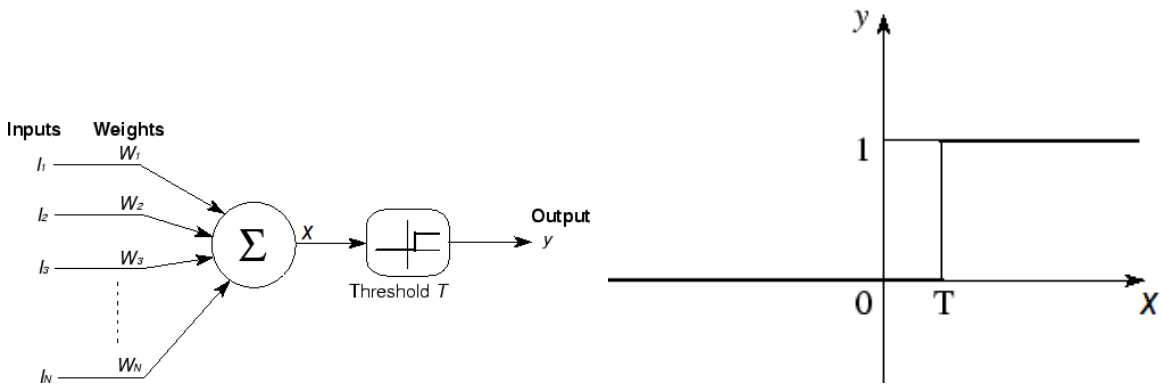
2.3 Umělý neuron

Umělé neurony jsou základní výpočetní jednotky neuronových sítí. Běžný umělý neuron se skládá z několika vstupů, váhových proměnných, výstupu a aktivační funkce. Hodnoty na vstupech mohou být generovány dalšími neurony, nebo mohou přicházet od vnějšího zdroje. Každý vstup má přiřazenu svou váhovou proměnnou (weight). [8]

Prvním modelem umělého neuronu byl McCulloch-Pitts (MP) neuron, nazývaný také jako Linear Threshold Unit (LTU), představený v roce 1943. Schéma tohoto neuronu je zobrazeno v obrázku 3a. Tento neuron zpracovává set vstupů a generuje jeden výstup y. Neuron klasifikuje výstupy do dvou možných tříd - výstup je tedy binární. Funkce neuronu se dá matematicky popsat následujícími vztahy (1). [9]

$$\begin{aligned} x &= \sum_{i=1}^{N} I_i W_i, \\ y &= f(x). \end{aligned} \tag{1}$$

Hodnoty I_x jsou vstupní hodnoty, W_x pak reprezentují váhy jednotlivých vstupů. Tyto hodnoty jsou normalizovány do intervalu $(0, 1)$ nebo $(-1, 1)$. Hodnota x reprezentuje součet všech vážených vstupů - potenciál neuronu. Funkce f je zobrazena na obrázku 3b. Hodnota T je prahová konstanta. Tato hodnota určuje při jaké hodnotě neuron sepne. [9]



(a) Schéma MP Neuronu

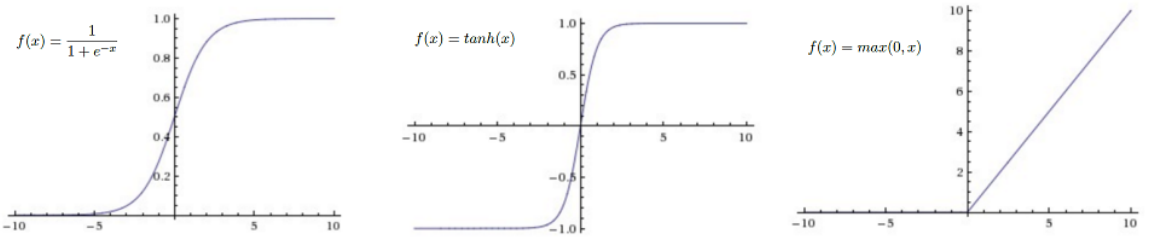
(b) Průběh prahové funkce neuronu.

Obrázek 3: Schéma a aktivační funkce MP neuronu. [9]

Modely dnes využívaných neuronů do značné míry vychází z MP modelu neuronu. Výpočet potenciálu probíhá stejným způsobem, k vstupům je však možné přidat ještě jeden dodatečný vstup - bias. Tento parametr potom slouží jako prahová hodnota. Výsledný potenciál lze vyjádřit vztahem (2):

$$x = bias + \sum_{i=1}^{N} A_i W_i. \tag{2}$$

Výsledný potenciál je poté předán aktivační funkci, která vygeneruje výstupní hodnotu neuronu. Existuje více typů aktivačních funkcí. Všechny využívané funkce jsou částečně lineární nebo nelineární. Na obrázku 4 jsou ukázány některé často používané aktivační funkce. Použitá aktivační funkce pak blíže definuje konkrétní vlastnosti modelu neuronu. [9, 10]



(a) Sigmoid (b) Tanh (c) ReLU

Obrázek 4: Používané aktivační funkce. [10]

2.4 Učení

Jednou z nejdůležitějších vlastností neuronových sítí (a strojového učení obecně) je jejich schopnost učit se. Neuronové sítě jsou adaptivní systémy, což znamená, že dokáží měnit svou vnitřní strukturu podle informací, které do nich proudí. Typicky se sítě adaptují upravováním váhových hodnot na vstupech jednotlivých neuronů. Existuje více přístupů k učení sítí. Vhodnost každého z přístupů závisí především na typu dat, se kterými se pracuje. [1]

- **Učení s učitelem (Supervised learning) -** Způsob učení, který vyžaduje spolu s trénovacími daty i očekávané správné výstupy pro jednotlivé vstupy. Při každém průchodu testovacích dat sití se výsledné hodnoty porovnávají s očekávanými hodnotami. Jsou-li hodnoty rozdílné, dochází k úpravám hodnot sítě.

  Tento způsob učení se dále dá rozdělit podle typu výstupu na dvě podskupiny - klasifikace a regrese. Při klasifikaci je na výstupu očekávána třída z kategorie - například jedna z hodnot "muž"a "žena". U regrese je výstupní hodnotou reálné číslo. Příkladem problému regrese je například vyhodnocení ceny nemovitosti.

- **Učení bez učitele (Unsupervised learning) -** Používá se v případech, kdy je hlavním účelem sítě vyhledávat vzory a struktury v datech nebo třídit data do předem neznámých skupin. Příkladem může být třídění zákazníků podle položek, které kupují. [1, 12]

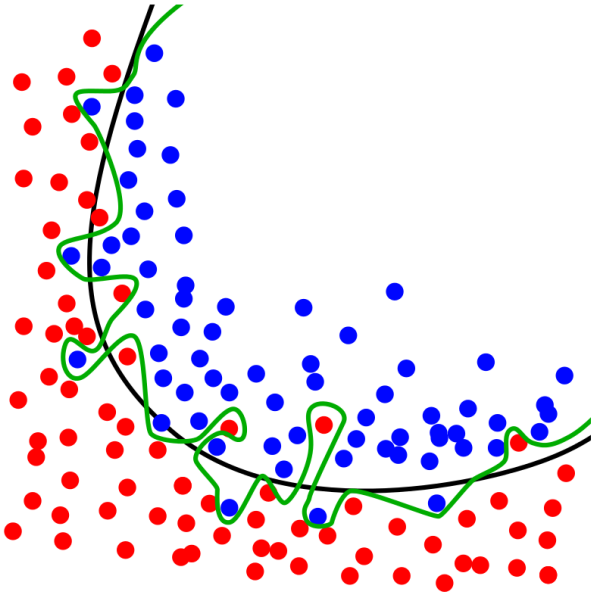
- **Učení zpětnou vazbou (Reinforcement learning) -** Strategie založená na pozorování, nejčastěji používaná v robotice nebo AI systémech. Učený systém (agent) se postupně učí chovat ve svém prostředí pomocí vykonávání akcí a sledování jejich výsledků. [1, 13]

Metody učení s učitelem se u neuronových sítí dají rozdělit do dvou skupin - gradientní a negradientní metody. Gradientní metody jsou metody využívající algoritmus gradientního sestupu

(gradient descent) pro minimalizaci chybových výsledků sítě. Typickým příkladem takovéto metody je algoritmus zpětného šíření (Backpropagation - BP). Mezi negradientní metody se pak řadí především metody využívající evolučních algoritmů, jako například GNARL nebo hyperNEAT. [11]

Důležitým parametrem mnoha algoritmů pro učení je **míra učení** (learning rate). Tato hodnota slouží k ovlivnění velikosti změn ve váhových hodnotách. Nízká hodnota tohoto parametru může zapříčinit velmi pomalý průběh učení. Příliš vysoká hodnota může naopak způsobit, že váhové hodnoty sítě budou oscilovat kolem optimálních hodnot, namísto aby se jim přibližovaly. Některé algoritmy umožňují změnu hodnoty tohoto parametru v průběhu učení. Takovýto přístup vede ve většině případů k urychlení celého procesu učení. [1, 14]

Přeučení sítě (anglicky Overfitting) je jev, který nastává, když se síť naučí trénovací množinu příliš těsně a špatně zobecňuje. Přeučení často nastává v případech, kdy je učená síť příliš komplexní pro řešený problém. Opakem přeučení je pak podučení (underfitting), které může nastat, když učená síť nemá dostatatek neuronů, aby dokázala řešení daného problému dostatečně reprezentovat. Obrázek 5 zobrazuje příklad přeučení. Přeučený model sice lépe mapuje všechny trénovací hodnoty, pro nové neznámé hodnoty ale pravděpodobně bude generovat výsledky s vyšší mírou chyby než černě znázorněný model. [14, 15]



Obrázek 5: Ilustrace přeučení. Černý průběh reprezentuje vhodně naučený model, zelený znázorňuje přeučený model [16].

Jednou z používaných metod pro prevenci přeučení sítě je využívání tzv. Dropout techinky. Dropout je založen na ignorování některých náhodně vybraných neuronů při trénování sítě. Síť tak může mít při každém trénovacím průchodu trochu jinou reálnou topologii, díky čemuž je

nucena více rozprostírat své váhy. Tento přístup snižuje šance, že chybové hodnoty jednotlivých neuronů v síti se navzájem vyruší. [17]

2.4.1 Backpropagation

Backpropagation je algoritmus pro úpravu váhových hodnot založený na výpočtu gradientu. Tato metoda byla vytvořena na počátku 70. let ve vícero vědeckých komunitách, v kontextu neurálních sítí je však používána až od roku 1985. V současnosti je tento algoritmus jedním z nejpopulárnějších algoritmů pro řešení optimalizace v neuronových sítích. [18]

Algoritmus BP je možné rozdělit do několika částí. V první části dochází k běžnému (dopřednému) průchodu sítí a výpočtu odhadované výstupní hodnoty pro daná vstupní data. Tato hodnota je následně porovnána s očekávanou hodnotou výstupu a je vypočtena chyba. Pro výpočet chyby se využívá cenová funkce (loss/cost/error function). Cílem celého algoritmu BP je optimalizace této cenové funkce. Jednou z často používaných cenových funkcí je kvadratická cenová funkce (Mean squared error), která je dána následujícím vztahem:

$$C(w,b) = \frac{1}{2n}\sum_{x}||y(x)-a||^2, \tag{3}$$

kde w a b jsou kolekce všech váhových a bias parametrů, n je celkový počet trénovacích příkladů, x je vstupní vektor, $y(x)$ je očekávaný výstup pro vstup x a $a(x)$ je odhadovaný výstup pro vstup x. Tato funkce se dá vyjádřit i pro jednotlivé trénovací příklady vztahem:

$$C_x = \frac{1}{2}||y-a||^2. \tag{4}$$

K úpravě vah v sítí je třeba pro každý neuron vypočíst jeho chybovou hodnotu δ_j^l (l je číslo vrstvy a j je číslo neuronu ve vrstvě). Pro výpočet této hodnoty ve výstupní vrstvě platí následující vztah:

$$\delta_j^L = \frac{\delta C}{\delta a_j^L}\sigma'(z_j^L), \tag{5}$$

kde L je celkový počet vrstev sítě, funkce σ představuje aktivační funkci a z je vážený vstup neuronu. Pro výpočet chybové hodnoty δ v ostatních vrstvách sítě se pak využívá vztah:

$$\delta_j^l = \sigma'(z^l)\sum_{k} w_k^{l+1}\delta_k^{l+1}, \tag{6}$$

kde w je váhový parametr neuronu a k reprezentuje všechny neurony ve vrstvě. Při výpočtu chybových hodnot δ se postupuje ve směru od výstupní vrstvy ke vstupní, jelikož pro výpočet chyby vrstvy l je potřeba znát chybovou hodnotu z vrstvy $l+1$. Pro následné výpočty nových hodnot vah slouží tento vzorec:

$$\frac{\delta C}{\delta w} = a_{in}\delta_{out}, \tag{7}$$

kde a_{in} je výstup aktivační funkce vstupního neuronu váhy w a δ_{out} je chybová hodnota neuronu pod který w spadá. [14, 19]

2.5 Perceptron

Perceptron je nejjednodušším modelem dopředné neuronové sítě, která je tvořena pouze jediným neuronem. Model sítě byl vynalezen Frankem Rosenblattem v roce 1957 v Cornell Aeronautical Laboratory. Perceptron je tvořen jedním nebo více vstupy, procesorem a jedním výstupem. Každý vstup má přiřazenou svou váhovou hodnotu, což je obvykle hodnota v intervalu $< -1, 1 >$.

Pro vygenerování výstupu perceptronu se nejdříve na vstupech přiřadí vstupní hodnoty, na které se následně aplikují jejich váhy. Vážené vstupy se dále sečtou a výsledná suma slouží jako vstup pro aktivační funkci. Výsledkem aktivační funkce je potom výstupní signál perceptronu. Výstupní signál je binární, může tedy nabývat pouze jedné ze dvou hodnot. [1, 2]

Pro úpravy váhových parametrů při učení se využívá následující algoritmus. Perceptronu jsou předány vstupní hodnoty, ze kterých perceptron vypočte odhadovaný výstup. Z odhadovaného ($GUESS$) a očekávaného ($DESIRED$) výstupu se vypočte hodnota chyby podle vzorce:

$$ERROR = DESIRED - GUESS. \tag{8}$$

Výsledná hodnota $ERROR$ potom slouží pro výpočet změny ve váhových hodnotách podle vzorce:

$$\Delta WEIGHT = ERROR * INPUT * LEARNING\_CONST. \tag{9}$$

Parametr $LEARNING\_CONST$ reprezentuje míru změny ve váhových hodnotách. Vysoká hodnota tohoto parametru znamená, že změny ve váhách budou více drastické. Takové chování může celý proces učení urychlit, může však dojít i k situacím, kdy příliš velká změna "přestřelí"optimální hodnotu váhy. Nízká hodnota pak umožňuje dosáhnout vyšší přesnosti, ale na úkor celkově delší doby učení sítě. [1]

Parametr $INPUT$ je hodnota vstupu, jehož váhový parametr je upravován. $\Delta WEIGHT$ představuje změnu váhového parametru. K úpravě váhového parametru se $\Delta WEIGHT$ přičítá k stávající hodnotě váhy (10), čímž vzniká nová hodnota váhového parametru NEW_WEIGHT:

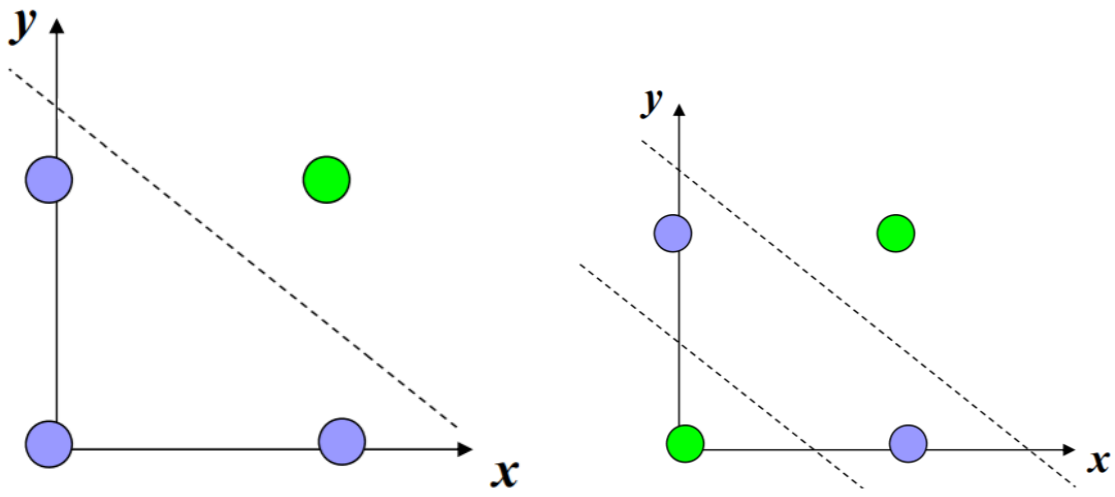
$$NEW\_WEIGHT = WEIGHT + \Delta WEIGHT. \tag{10}$$

Pro perceptron se dvěmi vstupy může parametr chyby nabývat pouze 3 různé hodnoty. Pokud byl výstup odhadnut správně a je tedy stejný jako očekávaná hodnota, je výsledná hodnota chyby nulová. Pokud byl očekáván výstup -1 a odhad $+1$, je chybová hodnota rovna -2 a pokud byl očekáván výstup 1 a odhadovaná hodnota je -1, chybová hodnota je rovna 2. Možné hodnoty chyby jsou uvedeny v tabulce 1. Tabulka vychází z předpokladu, že výstupní hodnoty perceptronu mohou být pouze $+1$ a -1. [1]

Desired	Guess	Error
-1	-1	0
-1	+1	-2
+1	-1	2
+1	+1	0

Tabulka 1: Tabulka možných chyb [1].

Perceptron zvládá řešit pouze lineárně separovatelné problémy. Lineárně separovatelný problém je takový problém, kdy je možné množinu dat vhodně rozdělit lineární funkcí. Příkladem lineárně separovatelných problémů jsou například logické funkce AND nebo OR. Nelineární problém je pak například funkce XOR. Na obrázcích 6 jsou příklady obou problémů. Pro řešení tohoto typu problému se dají využít více vrstvé sítě. [1]



(a) Lineárně separovatelná funkce AND

(b) Nelineárně separovatelná funkce XOR

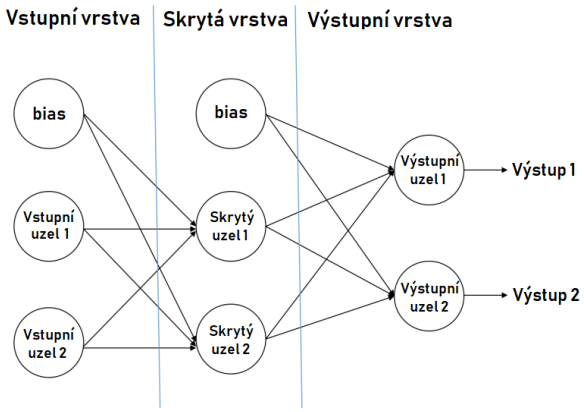
Obrázek 6: Ukázky lineárně a nelineárně oddělitelných problémů. Převzato z [20]

2.6 Vícevrstvé neuronové sítě

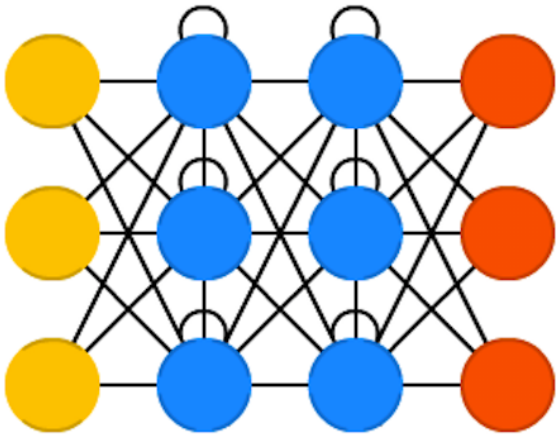
Vícevrstvé neuronové sítě (MLP) jsou nejpoužívanějším typem neuronových sítí. Využívají se při učení s učitelem a slouží pro řešení klasifikačních i regresivních problémů. Jako aktivační funkce se často používá sigmoida nebo ReLU. Základní metodou pro učení sítě je metoda zpětného šíření (viz. sekce 2.4.1), existují však i jiné metody jako třeba metoda sdružených gradientů. Mezi největší nevýhody těchto sítí patří obtížné řešení problému lokálních minim a poměrně dlouhá délka učení. [20]

Neurony jsou ve vícevrstvých sítích organizovány do vrstev. Spojení se v síti nacházejí jen mezi neurony ze sousedních vrstev. Vrstvy se dělí na tři základní typy: vstupní, skryté a výstupní.

Na vstupní vrstvě nedochází k žádným dodatečným výpočtům, vrstva slouží pouze pro předávání vnějších informací do sítě. Výstupní vrstva je zodpovědná za výpočty a převod informací mimo síť. Skryté vrstvy jsou pak všechny vrstvy mezi vstupní a výstupní vrstvou. Na obrázku 7a je zobrazeno obecné schéma MLP sítě. [10]



(a) Schéma vícevrstvé neuronové sítě [10].



(b) Schéma rekurentní neuronové sítě [4].

Obrázek 7: Topologie jednotlivých typů neuronových sítí.

2.7 Rekurentní neuronové sítě

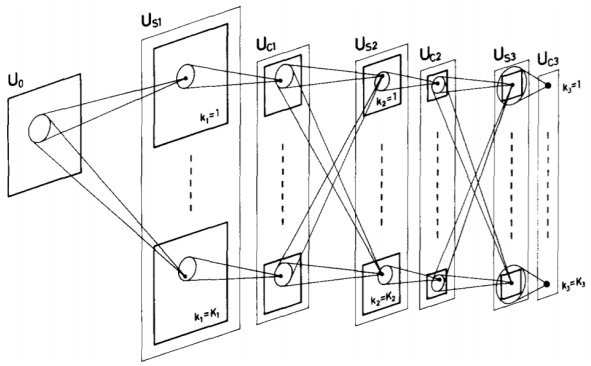
Rekurentní neuronové sítě (RNN) vycházejí z dopředných sítí. Od typických dopředných sítí se liší v tom, že nejsou bezestavové. Neurony v těchto sítích tak nepracují pouze s vstupy z předchozích vrstev, ale také s vnitřními stavy z přechozích iterací. Tato vlastnost jim umožňuje pracovat a učit se i vztahy z hlediska času. Tento typ sítí v poslední době nalézá uplatnění při řešení mnoha problémů jako například rozpoznávání řeči nebo překlad. Nevýhodou těchto sítí je jejich náročnost na trénování. Obecné schéma RNN je zobrazeno na obrázku 7b. [4]

3 Konvoluční neuronové sítě

Konvoluční neuronové sítě (CNN) jsou biologicky inspirovanou variantou vícevrstvých neuronových sítí, které se využívají pro zpracování dat s mřížkovitou topologií, jako jsou například obrázky. Princip funkčnosti těchto sítí vychází z poznatků z výzkumu vizuálního vnímání živočichů. Tyto sítě se typicky používají v odvětvích jako počítačové vidění nebo zpracování přirozeného jazyka. [21, 22]

Historické kořeny konvolučních neuronových sítí lze sledovat až do 60. let minulého století, kdy D. Hubel a T. Wiesel provedli výzkum [23] zaměřený na kočičí vizuální kortex. V rámci tohoto výzkumu byly kočce do části mozku zavedeny elektrody a následně byly před kočku promítány světelné vzory. Ve výzkumu bylo zjištěno, že jednotlivé buňky reagují pouze na malé regiony pole vidění. Tyto vjemové regiony (angl. receptive fields) jsou organizovány tak, aby pokrývaly celé pole vidění. Dále byly identifikovány dva typy buněk- jednoduché a komplexní kortikální buňky. Jednoduché buňky silně reagují na specifické vzory hran ve svém vjemovém regionu. Komplexní buňky pak reagují na stejné vzory, ale mají větší vnímané pole a jsou méně citlivé na pozici vzoru. [25, 26]

Prvním modelem sítě, který byl simulován na počítači byl Neocognitron [27]. Model byl navržen K. Fukushimou v roce 1980 a vychází z práce Hubela a Wiesela. Síť je sestavena z S-buněk a C-buněk, které jsou podobné jednoduchým a komplexním kortikálním buňkám. Schéma sítě je zobrazeno na obrázku 8. Tento model je považován za předchůdce dnešních konvolučních neuronových sítí. [22, 27]



Obrázek 8: Schéma sítě neocognitronu zachycující propojení vrstev. [27]

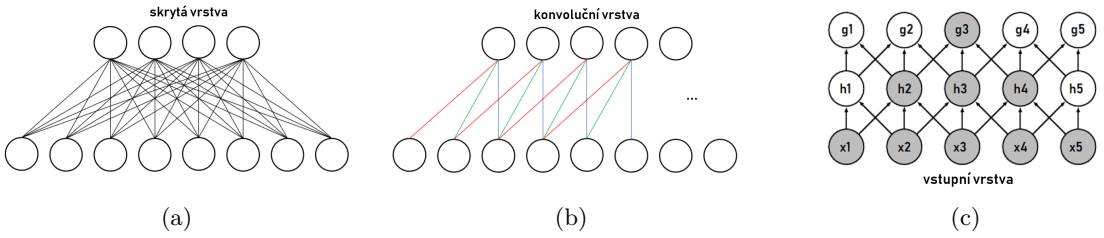
Obecná architektura konvolučních neuronových sítí se v několika ohledech liší od architektury MLP sítí. Základním prvkem těchto sítí jsou **konvoluční vrstvy**, podle nichž jsou taky sítě pojmenovány. Dalšími typy vrstev v síti jsou podvzorkovací (pooling) vrstvy a vrstvy nelinearity. Sítě jsou obvykle tvořeny řadou konvolučních a vzorkovacích vrstev, na které případně navazují plně propojené vrstvy (běžné skryté vrstvy z MLP sítí). Obecné schéma konvoluční neuronové sítě je ilustrováno na obrázku 12. [22, 28]

3.1 Konvoluční vrstva

Tento typ vrstev je základním stavebním blokem CNN. Vrstva se skládá z naučitelných kernelů (filtrů), které slouží pro zachytávání rysů ze vstupních dat. Každá jednotka v této vrstvě zpracovává vstup pouze z malého regionu z předchozí vrstvy. Tento region se nazývá vjemové pole (receptive field). Při dopředném průchodu sítí se pro každý filtr provádí operace konvoluce, jejíž výsledkem je mapa rysů (feature map). Výstupem konvoluční vrstvy jsou pak všechny takto vygenerované mapy. [22]

Využití konvoluce s sebou nese několik důležitých vlastností, jako řidké propojení vrstev nebo sdílení parametrů. Řidké propojení (angl. sparse connectivity nebo sparse weights) je způsob propojení sousedících vrstev. Namísto úplného propojení mezi vrstvami (dense) je každá jednotka propojena jen s malým regionem z předchozí vrstvy. Tento přístup zajišťuje konvolučním vrstvám mnohem menší počet parametrů a nižší výpočetní náročnost. Zároveň s rostoucím počtem konvolučních vrstev mohou jednotky v hlubších vrstvách nepřímo reagovat na čím dál větší vstupní region. [21, 29]

Porovnání mezi řidkým a úplným propojením je vyobrazeno na obrázcích 9a a 9b. Na obrázku 9c je pak zobrazeno nepřímé vjemové pole vyšších vrstev sítě.



(a) (b) (c)

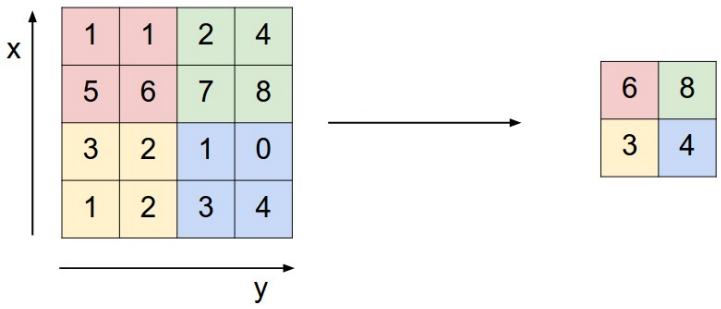
Obrázek 9: Ilustrace vlastností konvolučních vrstev: a) představuje úplné propojení vrstev [29]; b) představuje řidké propojení, shodné barvy hran reprezentují sdílené parametry [29]; c) šedé jednotky vstupní vrstvy představují rozsah nepřímého vjemového pole neuronu z vyšší vrstvy [21].

Sdílení parametrů umožňuje využití stejného parametru na více místech v modelu. V běžné neuronové síti je každý váhový parametr při výpočtu výstupu sítě použit právě jednou. V konvolučních vrstvách CNN však parametry tvoří filtry, které se aplikují na vstupní data. U této operace dává větší smysl, aby byl stejný filtr použit na každou vstupní pozici. Tato vlastnost nijak neovlivňuje rychlost výpočtu výsledku sítě, ovlivňuje však množství parametrů a tím i množství paměti potřebné pro uložení modelu. Sdílení parametrů je naznačeno v obrázku 9b. [21]

3.2 Pooling vrstva

Poolingové vrstvy slouží pro transformaci vstupních dat do lépe použitelné podoby tak, aby důležité informace zůstaly zachovány a nedůležité detaily byly odfiltrovány. Vrstvy se používají pro podvzorkování dat, čímž dojde ke snížení počtu parametrů a množství výpočtů v síti.

Vrstva prochází vstupní data po blocích, přičemž z každého bloku generuje jedinou hodnotu. Existují dvě základní formy poolingu - pooling průměrem a maximální hodnotou (average a max-pooling). Používanější variantou je pooling maximální hodnotou (ukázka na obrázku 10), kdy jako výsledná hodnota bloku je vybrána vždy nejvyšší hodnota. [30, 35]

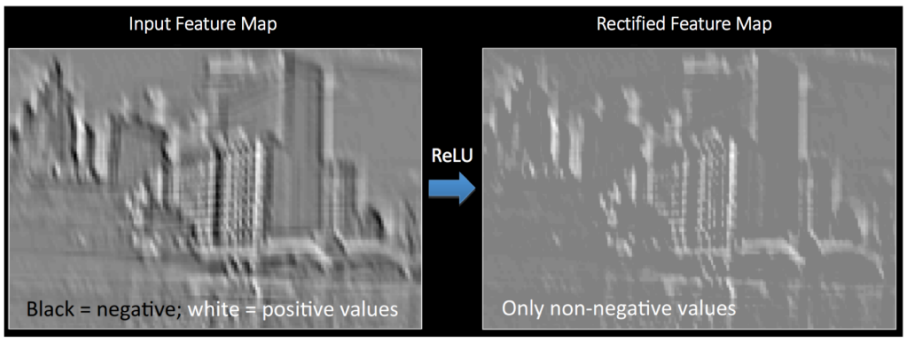


Obrázek 10: Ukázka funkce max-poolingu s rozměry 2×2 a střídou 2 [35].

3.3 Nelineární vrstva

Nelineární vrstvy jsou vrstvy tvořené neurony, které slouží pro aplikování aktivačních funkcí na data v síti. Hlavní funkcí těchto vrstev je zavádění nelinearit do dat. Tato funkce je v CNN potřeba, jelikož konvoluční vrstvy samy o sobě žádnou nelinearitu nezavádějí (konvoluční operace jsou lineární operace). Z tohoto důvodu jsou ve většině modelů CNN konvoluční vrstvy následovány nelineární vrstvou.

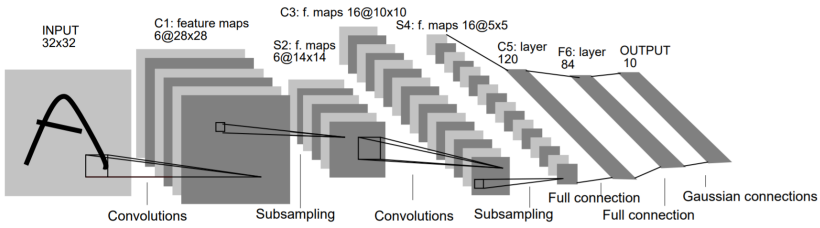
Nejčastěji používanými aktivačními funkcemi bývají sigmoid, tanh nebo ReLU (viz. obrázky 4). Patrně nejpreferovanější z těchto funkcí je ve většině případů funkce ReLU. To je způsobeno především faktem, že neuronové sítě využívající tuto aktivační funkci se trénují několikrát rychleji. Na obrázku 11 je ukázána aplikace nelineární vrstvy s ReLU aktivací na mapu příznaků. Vstupní mapa obsahuje kladné i záporné hodnoty, výstupní mapa již jen kladné. [22]



Obrázek 11: Ukázka aplikace ReLU operace na mapu příznaků. Záporné hodnoty jsou znázorněny černě. Převzato z [24].

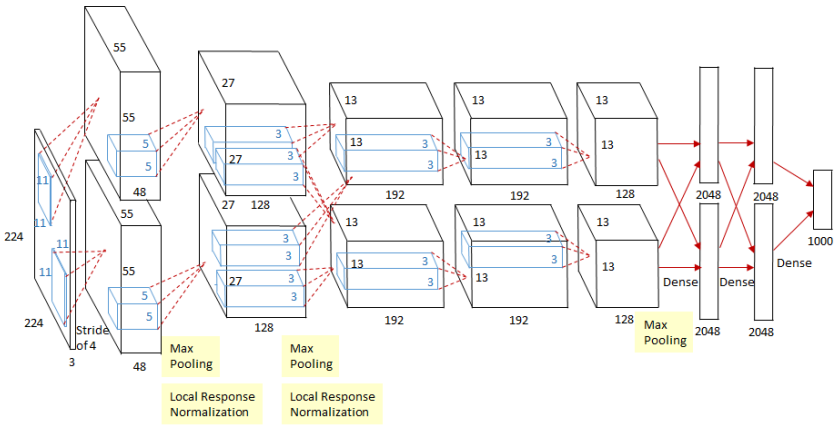
4 Modely konvolučních neuronových sítí

Jako první úspěšná architektura moderních CNN je považován **LeNet-5**. Tato architektura vznikla v roce 1998 v týmu pod vedením Yanna LeCuna [25] a dodnes představuje základ designu většiny konvenčních CNN. Síť byla určena pro klasifikaci ručně psaných číslic a pro učení využívala algoritmus BP. Původní síť (schéma na obrázku 12) byla tvořena sedmi výpočetními vrstvami (nepočítaje vstupní vrstvu) a pro podvzorkování využívala average pooling. [22, 25]



Obrázek 12: Architektura sítě LeNet. Převzato z [25].

V roce 2012 byl Alexem Krizhevskym et. al. představen výrazně hlubší model konvoluční neuronové sítě - **AlexNet** [31]. Tento model ve stejném roce se značným náskokem oproti ostatním metodám vyhrál soutěž pro vizuální rozpoznávání objektů - ImageNet Large Scale Visual Recognition Challenge (ILSVRC). Výsledky této architektury způsobily průlom ve využívání strojového učení a počítačového vidění pro klasifikační úlohy a nárust zájmu o deep learning. [32]

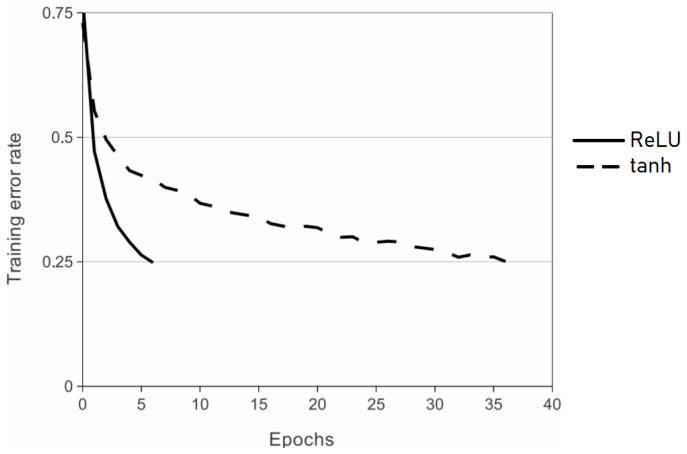


Obrázek 13: Architektura sítě AlexNet. Převzato z [33].

AlexNet je strukturou podobný architektuře LeNet, využívá ale větší množství filtrů na jednotlivých vrstvách. Celá architektura (zobrazeno na obrázku 13) se skládá z osmi vrstev. Kvůli limitům tehdejšího hardware byla architektura navržena pro využití dvou paralelních grafických karet. Trénování sítě bylo realizováno na dvou NVIDIA GTX 580 3GB GPU. [31, 32]

Jednou ze zásadních změn oproti předcházejícím architekturám je využití ReLU jako aktivační funkce neuronů. Neuronové sítě využívající ReLU namísto běžně užívané *tanh* se totiž

dokážou učit několikanásobně rychleji. Tento fakt je ilustrován na obrázku 14. Graf znázorňuje pokles chyby dvou ekvivalentních čtyřvrstvých CNN s rozdílnou aktivační funkcí na datasetu CIFAR-10. Plná čára reprezentuje síť využívající ReLU, přerušovaná pak reprezentuje *tanh*. Síť využívající ReLU v tomto případě šestkrát rychleji dosáhne poměru chyby 25%. [31]

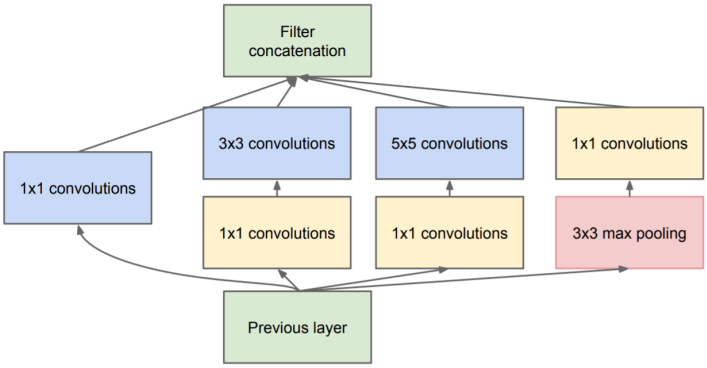


Obrázek 14: Ukázka poklesu chyby pro různé aktivační funkce. Převzato z [31].

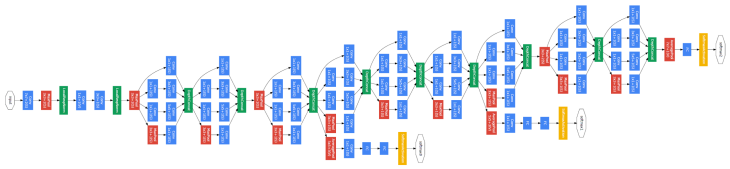
GoogleNet je architektura navržená Christianem Szegedy et. al. z Google Inc. [34]. V roce 2014 tato architektura zvítězila v ILSVRC. Architektura je navržená s úmyslem snižování výpočetní náročnosti v porovnání s běžnými CNN. Klíčovým prvkem v této architektuře jsou tzv. Inception moduly, díky kterým došlo k drastickému snížení parametrů v síti. [32, 35]

Incepční moduly se skládají z několika souběžných konvolučních vrstev s různými parametry a jsou zakončené propojením těchto vrstev zpět do jednoho toku. Použití více konvolucí s různou velikostí filtrů umožňuje síti hledat v datech rysy s lepší invariancí vůči jejich velikosti. Konvoluční vrstvy 1×1 pak slouží pro snížení objemu dat (a výpočetní náročnosti) před náročnějšími 3×3 a 5×5 konvolucemi. Schéma incepčního modulu použitého v architektuře GoogleNet je vyobrazen na obrázku 15. [34, 36]

Původní architektura je dohromady tvořena 22 vrstvami (obrázek 16), což je podstatně více než předchozí architektury. Celkový počet učitelných parametrů v síti je však v porovnání s AlexNetem mnohem menší. [32]

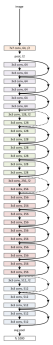


Obrázek 15: Schéma incepčního modulu [34].

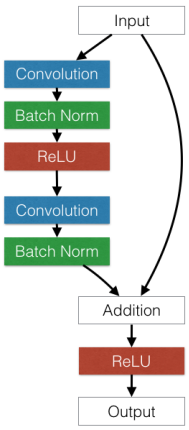


Obrázek 16: Architektura sítě GoogleNet [34]. Zvětšená verze je v příloze B

V roce 2015 vyhrála ILSVRC architektura **ResNet** (Residual Network) vyvinutá Kaimingem He et. al. [37]. Tato architektura představuje speciální spoje mezi konvolučními vrstvami - skokové spoje (skip/shortcut connections). Tyto skokové spoje pomáhají řešit problémy hlubokých sítí, kdy příliš velký počet vrstev sítě způsobuje zvýšení chybové hodnoty sítě. Obrázky 17 zobrazují základní schéma skokového spoje a schéma architektury ResNet. [35, 38]



(a) Architektura ResNet [37]. Zvětšená verze v příloze C.



(b) Základní schéma skokového spoje [38].

Obrázek 17: Architektura ResNet.

5 Metody detekce a klasifikace akcí

Rozpoznávání lidských akcí je důležitým tématem v oboru počítačového vidění. Tato disciplína nachází uplatnění v širokém spektru aplikací jako například digitální dozor a monitorování, ukládání a hledání videí, inteligentní rozhraní mezi lidmi a počítači nebo rozpoznávání identity. Cílem rozpoznávání akcí je ze vstupního videa správně vyhodnotit a klasifikovat akci. Systémy pro rozpoznávání akcí mohou být typicky rozděleny do dvou hlavních částí - část pro reprezentaci a část pro klasifikaci. Část pro reprezentaci se zabývá převodem vstupního videa na vektor příznaků (feature vector), který je dále použit v klasifikační části pro zařazení akce do jedné ze tříd. V poslední době se také často využívají hluboké neuronové sítě pro spojení těchto dvou komponent do jednotného trénovatelného systému. Metody pro rozpoznávání akcí musí v realitě řešit řadu problémů, jako například varianci ve vzhledu a rozměrech člověka nebo rozdílné způsoby provádění akcí. [39, 40]

5.1 Reprezentace akcí

Jedním z hlavních problémů u rozpoznávání akcí je jakým způsobem akci obsaženou ve vstupních datech vhodně reprezentovat. Jednotlivé záznamy akcí se mohou navzájem lišit v mnoha aspektech jako například rychlost pohybu, úhel pohledu kamery nebo rozdílnosti v póze a provedení akce. Úspěšné metody pro reprezentaci akcí by pak měly být efektivní jak z hlediska charakterizace akcí, tak i z pohledu výpočetní náročnosti.

Cílem metod pro reprezentaci akcí je extrakce důležitých informací ze vstupních dat, odfiltrování nedůležitých informací a minimalizace variací mezi shodnými akcemi. Tomuto procesu se také říká extrakce příznaků (feature extraction) a jeho výsledkem je vektor příznaků. Tento vektor je dále použitelný při následné klasifikaci akce. Mezi často používané metody extrakce příznaků patří například metody STIP, HOG nebo SSM. [39]

Příznaky je možné dělit podle více kritérií, jedním z nich je například dělení na globální a lokální příznaky. Globální příznaky slouží k popisu akce jako jednoho celku. Tyto metody obvykle nejdříve lokalizují region zájmu (region of interest - ROI) v obrázku a následně celý tento region zakódují do výsledného příznaku. Výhodou tohoto přístupu je množství informací obsažených ve výsledném příznaku. Nevýhodou je pak velká náchylnost k šumu. Úspěšnost také značně závisí na přenosti lokalizace ROI. [39, 41]

Lokální příznaky popisují data jako kolekci jednotlivých částí obrázku. Metody pro detekci lokálních příznaků nejdříve detekují významné body v obrázku a následně zakódují malé regiony z okolí těchto bodů. Výsledné části jsou pak zkombinovány do konečného příznaku. Tyto metody jsou méně náchylné k šumu a částečnému zakrytí některých částí. Kvůli nutnosti detekce potřebného množství bodů však někdy vyžadují určité předzpracování dat. [39, 41]

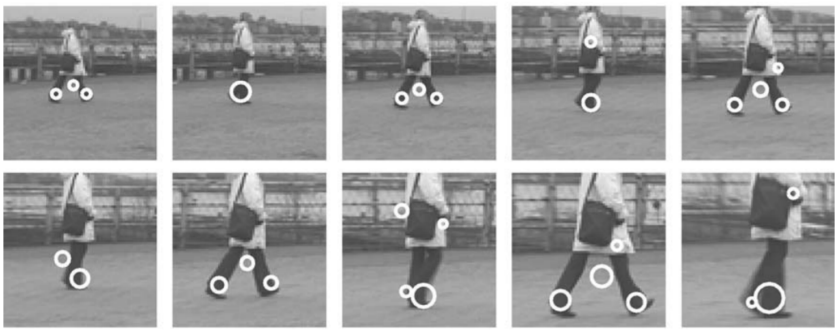
Další způsob dělení příznaků je založen na dimenzionalitě a typu dat, se kterými se pracuje. Běžné RGB kamery zachytávají reálný 3D svět pomocí projekce jako sérii 2D obrázků. Při tomto procesu nevyhnutelně dochází ke ztrátě některých důležitých 3D prostorových dat. Další

problémy potom způsobují i změny měřítka, úhlu pohledu nebo světelných podmínek. V posledních letech se však začíná nabízet i možnost využívání hloubkových senzorů. Pro tyto účely je využíván především systém Microsoft Kinect. V porovnání s RGB snímky mají hloubkové mapy výhodu, jelikož obsahují dodatečnou hloubkovou hodnotu. [42, 43]

V poslední době jsou také využívány metody založené na 3D pozicích kloubů. Tyto pozice jsou získávány využitím hloubkových map a algoritmů pro odhad kostry. Výhodou reprezentace založené na kostře je zejména odolnost vůči změnám úhlu pohledu, vzhledu a malá náchylnost k okolnímu šumu. [44]

5.1.1 Space-Time Interest Points

Časoprostorové zájmové body (STIP) jsou jednou z metod lokální extrakce příznaků z videa. Tato metoda byla vyvinuta Ivanem Laptevem v roce 2005 [45]. Metoda je založena na Harrisově detektoru rohů, který je rozšířen i pro aplikaci ve směru časové osy. Příznaky STIP jsou detekovány v regionech s vysokými rozdíly intenzity pixelů ve všech směrech (x, y, t). Metoda tak detekuje pouze rohy, které mění rychlost svého pohybu v průběhu času. STIP příznaky se typicky objevují v artikulovaných pohybech (chůze, běh, skoky), neobjevují se však u konstantího pohybu. [46, 47]



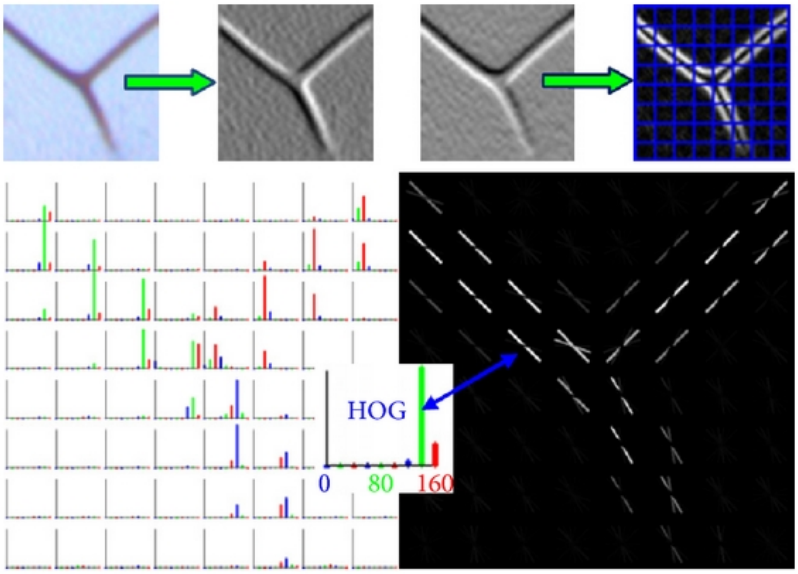
Obrázek 18: Ukázka detekce STIP z video sekvence lidské chůze [45].

5.1.2 Histogram of Oriented Gradients

Histogram orientovaných gradientů (HOG) je deskriptorem založeným na výpočtu gradientů. Základní myšlenkou za touto metodou je fakt, že vzhled a tvar objektu v daném místě se často dá vhodně popsat distribucí lokálních gradientů intenzity nebo směrů hran. Tato metoda je v praxi implementována v následujících krocích. Nejdříve je obrázek rozdělen do malých regionů - buněk. V rámci každé buňky je následně vytvořen histogram z gradientů směru nebo orientace hrany všech pixelů v buňce. Tyto histogramy tvoří výsledný přiznak. [48]

Pro menší náchylnost deskriptoru ke změnám ve stínech a osvětlení ještě dochází k normalizaci kontrastu v jednotlivých buňek. Tohoto lze docílit vypočítáním síly histogramu v rámci

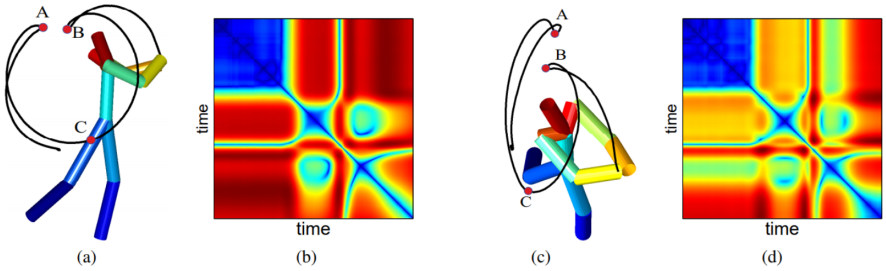
většího bloku v okolí buňky. Tato hodnota je následně použita pro normalizaci všech buněk v daném bloku. Na obrázku 19 je vyobrazen příklad HOG deskriptoru. [48]



Obrázek 19: Ukázka metody HOG. Vrchní řada obrázků zachycuje výpočet gradientu a segmentace obrázku do buněk. Spodní obrázky potom představují výsledné lokální histogramy a jejich grafickou reprezentaci. Převzato z [49].

5.1.3 Self-similarity Matrix

Self-similarity matrix (SSM) využívá pro reprezentaci akce speciální matici sestavenou ze vzdáleností mezi nízkoúrovňovými příznaky jednotlivých snímků akce. Pro sekvenci snímků $I = \{I_1, I_2, ...I_T\}$ je výsledná matice symetrickou maticí s rozměry $T \times T$. Hodnota v matici d_{ij} pak reprezentuje vzdálenost mezi nízkoúrovňovými příznaky snímků I_i a I_j. Jednotlivé hodnoty v matici popisují vzdálenost příznaků. Diagonála matice sestává z 0, jelikož je zde každý snímek porovnáván sám se sebou. Výpočet vzdálenosti záleží na zvoleném typu příznaků. Při využití kloubů může být výsledná vzdálenost vypočítána jako průměr euklidovské vzdálenosti jednotlivých kloubů. Ukázka SSM je zobrazena na obrázku 20. Výhodou této metody je především její nezávislost na úhlu pohledu u zpracovávaných dat. [50]



Obrázek 20: Ukázka pohledové nezávislosti metody SSM. Obrázky a) a c) ukazují pozice kloubů pro akci "golfový úder"zaznamenané z rozdílných úhlů. b) a d) reprezentují SSM jednotlivých snímků. Z matic jde vidět, že i přes rozdílné pohledy je struktura obou matic velmi podobná. Převzato z [50].

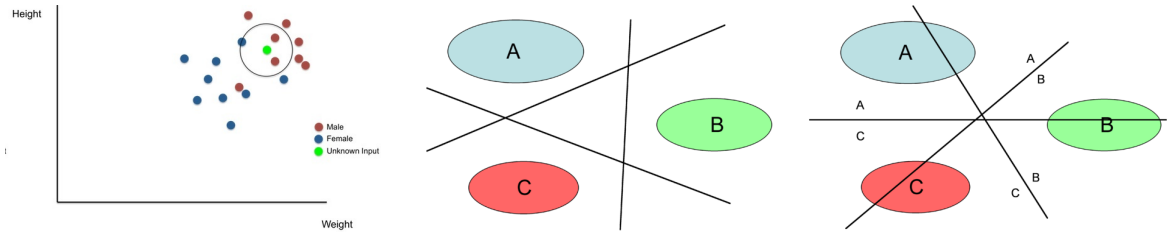
5.2 Klasifikace akcí

Po vyhodnocení reprezentací ze vstupních dat se z problému rozpoznávání akcí stává klasifikační problém. Pro klasifikaci je nejdříve nutné z trénovacích dat naučit klasifikátor rozpoznávat hranice pro jednotlivé třídy akcí. Existuje celá řada algoritmů pro klasifikaci. Vhodnost jednotlivých algoritmů je závislá na nátuře zpracovávaných dat, zvolené reprezentaci akcí i velikosti trénovacího setu. Mezi běžně využívané klasifikátory se řadí například metody K-NN nebo SVM. [41]

5.2.1 K-Nearest Neighbor

Klasifikátor K-tý nejbližší soused (K-NN) pro klasifikaci využívá vzdálenost mezi hledanými daty a záznamy z trénovacích dat. Nová data jsou klasifikováná podle třídy s nejvyšším počtem výskytů mezi k nejbližšími trénovacími záznamy. Výpočetní náročnost tohoto přístupu je závislá na množství trénovacích dat. Alternativně může být pro každou třídu akce vypočten průměrný prototyp. Klasifikace tímto algoritmem může být prováděna buď pro každý snímek akce zvlášť nebo pro všechny snímky najednou. Při klasifikaci s využitím všech snímků se musí dodatečně všechny akce zarovnat na jednotný počet snímků. [41]

Na obrázku 21a je zobrazen příklad klasifikace pomocí K-NN. Graf zachycuje klasifikaci pohlaví podle výšky a váhy osoby s parametrem $k = 3$. Vstupní data v tomto případě budou klasifikována jako muž.



(a) Příklad klasifikátoru K-NN [51]. (b) SVM - jeden proti všem. [52]. (c) SVM - jeden proti jednomu. [52].

Obrázek 21: Příklady klasifikačních algoritmů.

5.2.2 Support Vector Machines

Support Vector Machine (SVM) je algoritmem strojového učení, ve kterém je klasifikace prováděna na základě lineárních nadrovin. Při trénování dochází k umístění těchto nadrovin v prostoru příznaků tak, aby rozdělovaly jednotlivá trénovací data s rozdílnými značkami. Pokud nejsou trénovací data lineárně separovatelná, využívá se kernelové funkce. Tato funkce transformuje data do vyšší dimenze, ve které jsou lineárně separovatelné. [53]

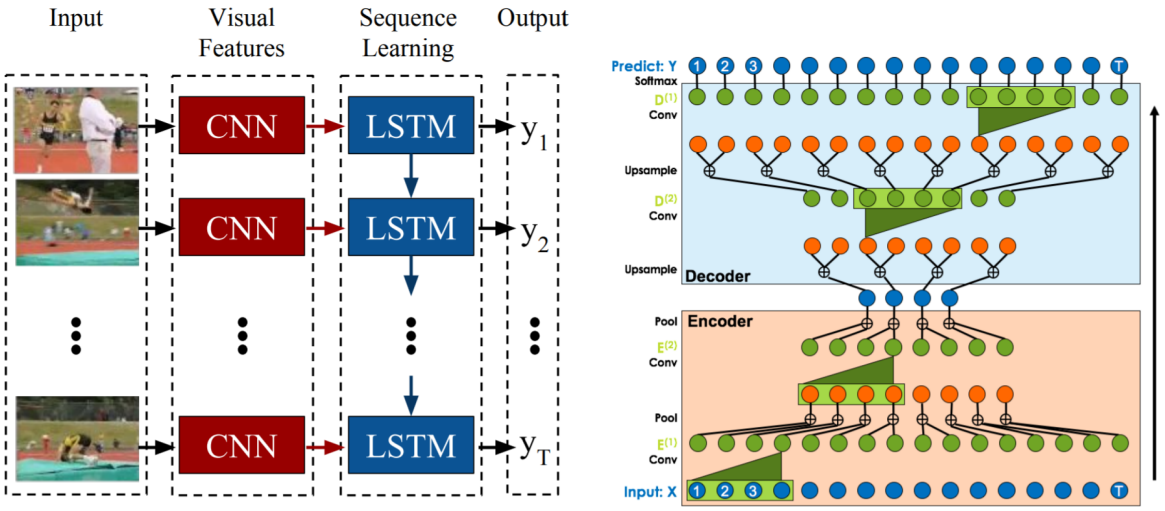
Základní verze SVM je určena pouze pro binární klasifikaci, metoda je však rozšiřitelná i pro klasifikaci do více tříd. Existují dva způsoby jak dosáhnout vícetřídové klasifikace, oba jsou založené na kombinaci více binárních SVM. Při řešení jeden proti všem (angl. one-against-all) kombinuje binární SVM tak, že každá instance porovnává, jestli daný vstup patří nebo nepatří do jedné ze tříd. Tento způsob vyžaduje N binárních SVM pro klasifikaci do N tříd. Řešení jeden proti jednomu (one-against-one) pak využívá SVM pro vyhodnocení dvou sousedních tříd. Tento způsob je přesnější, ale mírně pomalejší. Vyžaduje $\frac{N(N-1)}{2}$ binárních SVM. Obrázky 21b a 21c ukazují rozdělení prostoru příznaků s využitím obou zmíněných metod. [52, 53]

5.3 Využití hlubokých architektur

I přes velkou úspěšnost ručně vytvořených příznaků mají tyto metody několik nevýhod. Vývoj takovýchto systémů pro extrakci příznaků vyžaduje dobré znalosti dané domény a spotřebovává lidský čas. Další nevýhodou je fakt, že takto vytvořené příznaky obvykle nejsou dost obecné pro použití na velkých datasetech. Díky současnému rozvoji hlubokých neuronových sítí se v poslední době velká část pozornosti upírá na učení příznaků s využitím metod deep learningu. Nedávné modely hlubokých neuronových sítí dosahují překvapivě vysoké přesnosti na širokém spektru akčních datasetů. Mezi používané DL metody pro rozpoznávání akcí patří například metoda dlouhodobých rekurentních konvolučních sítí (LRCN) nebo Temporální konvoluční sítě (TCN). [39]

Long-term recurrent convolutional networks (LRCN) kombinují konvoluční neuronové sítě pro extrakci příznaků s rekurentními neuronovými sítěmi. Tento přístup síti umožňuje zpracovávat časové závislosti mezi příznaky. Rekurentní část sítě využívá LSTM (long short-term

memory) jednotky. Tyto jednotky umožňují v určitých případech zapomenout předchozí vnitřní stavy. Temporal convolutional networks (TCN) v současností dosahují lepších výsledků než sítě založené na LSTM . Tento typ sítí využívá pro zachytávání časových příznaků temporální konvoluční vrstvy, poolingů a nadvzorkování. Oba druhy sítí jsou znázorněny na obrázcích 22 [54, 55]



(a) Základní struktura LRCN sítě. [54].

(b) Struktura TCN sítě. [55]

Obrázek 22: Struktury LRCN a TCN sítí.

6 Implementace konvoluční neuronové sítě

K implementaci a následnému porovnání byly vytvořeny dva modely konvolučních neuronových sítí. První model je založen na architektuře LeNet a druhý na architektuře GoogLeNet. Pro implementaci modelů konvolučních neuronových sítí byl vybrán programovací jazyk Python. Tento jazyk byl vybrán především díky snadné manipulaci, rychlému vývoji, přenositelnosti a neposledně také kvůli podpoře dále použitých knihoven a frameworků. K organizaci modulů a prostředí v tomto jazyce byly využity nástroje virtualenv a pip.

K vytvoření, trénování a otestování modelů byla použita knihovna Keras v kombinaci s frameworkem Tensorflow běžícím na GPU díky technologii CUDA. K formátování vstupů modelů byla použita knihovna OpenCV. Při trénování a následném testování byly použity datasety UTKinect-Action3D a MSR Action3D. K urychlení procesu trénování byl využit školní server merlin1.

Pro snadnější orientaci v projektu byly jednotlivé logické části rozděleny do několika složek. V kořenové části projektu se nalézají centrální skripty, které zprostředkovávají veškeré ovládání. Tyto centrální skripty pak volají programy z jednotlivých adresářů v projektu, které dále obsahují konkrétní funkcionalitu. Skripty pro tvorbu modelů sítí se nacházejí ve složce *models/*, ostatní skripty jsou pak organizovány do složek *MSR/* a *UTKinect/*. Struktura projektu je detailněji popsána v příloze A.

6.1 Keras

Keras [56] je knihovna napsaná v jazyce Python, která poskytuje vysokoúrovňové rozhraní pro práci s neuronovými sítěmi. Cílem Kerasu je usnadnění a urychlení vývoje a experimentace, čehož se snaží docílit zaměřením na uživatelskou přívětivost, modularitu a snadnou rozšiřitelnost.

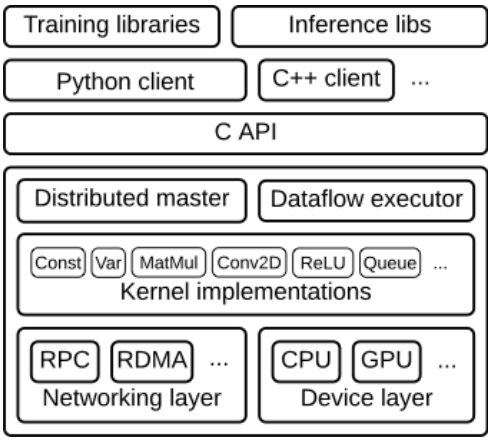
Knihovna sama o sobě pouze dodává jednotné rozhraní. Pro práci vyžaduje další framework, přes který pak dále funguje. V současnosti Keras podporuje 3 frameworky, nad kterými může pracovat - Tensorflow, Theano a Microsoft Cognitive Toolkit (CNTK).

6.2 Tensorflow

Tensorflow [57] je open-source framework určený pro numerické výpočty využívající grafy a pro vývoj aplikací využívajících metody strojového učení. Framework byl vyvinut týmem Google-Brain od společnosti Google a původně byl určen pouze pro interní práci a výzkum v rámci společnosti. V roce 2015 byl vypuštěn na veřejnost pod licencí Apache 2.0.

Framework v současnosti podporuje práci v jazycích C++, Python a Javascript a pro práci dokáže využívat i GPU. Google rovněž vyvíjí speciální procesorové jednotky (TPU) určené pro strojové učení a optimalizované pro tensorflow [58].

Nejnovější stabilní verzí frameworku je verze 1.13, pracuje se však na verzi 2.0, která přináší řadu změn včetně odstraňování zastaralých komponent frameworku. Tensorflow 2.0 je momentálně dostupný pouze jako alpha verze.



Obrázek 23: Obecná architektura frameworku TensorFlow. Vrstva C API odděluje uživatelský kód v různých jazycích od jádra frameworku. Převzato z [59]

6.3 OpenCV

OpenCV [60] je open-source knihovna obsahující širokou škálu optimalizovaných funkcí zaměřených převážně na počítačové vidění, zpracování signálů a strojové učení. Knihovna je multiplatformní a je dostupná pro jazyky C++, Python a Java.

OpenCV vznikla již v roce 2000 v rámci výzkumné divize společnosti Intel. V roce 2012 vývoj knihovny přešel pod neziskovou organizaci OpenCV.org. Nejnovější verze knihovny je v době psaní této bakalářské práce verze 4.0. Knihovna je licencovaná pod BSD licencí, diky čemuž je bezplatně dostupná i pro komereční použití.

6.4 CUDA

CUDA (Compute Unified Device Architecture) je model platformy pro paralelní výpočty a programovací model vytvořený společností Nvidia. Dostupnost architektury je omezena pouze na grafické karty od společnosti Nvidia. Platforma umožňuje vývojářům využívat výkon grafických karet pro vlastní obecně zaměřené výpočty a operace. Tento přístup se také označuje jako GPGPU - General-Purpose computing on Graphics Processing Units.

Hlavní výhodou GPGPU oproti běžným výpočtům na CPU je především podpora masivního paralelismu. GPU v porovnání s CPU totiž zvládají zpracovávat mnohonásobně větší množství vláken najednou.

Programovací model platformy CUDA je dostupný v jazycích C, C++, Fortan, Python či MATLAB. Platforma CUDA vznikla v roce 2006. Současná nejnovější verze CUDA Toolkitu je verze 10.1. [61, 62]

(a) RGB snímek (b) Hloubková mapa (c) Souřadnice kloubů

Obrázek 24: Ukázka dat UTKinect datasetu v různých formátech

6.5 Datasety

Sítě byly otestovány na dvou datasetech. Prvním datasetem je **UTKinect-Action3D dataset** (dále jen UTKinect) [63]. Tento dataset byl vytvořen Lu Xia et al. a k zachycení bylo využito zařízení Kinect. Dataset je tvořen 10 typy akcí: chůze, sednutí, postavení se, zvednutí předmětu, nesení předmětu, hod, strčení, potáhnutí, mávání rukama a zatleskání. Byly shromážděny akce od celkem deseti subjektů, přičemž každá z akcí byla zaznamenána dvakrát za každý subjekt. Celkem tedy dataset obsahuje 200 zaznamenaných akcí.

Záznamy jsou rozděleny do sekvencí tak, jak byly pořizovány. Každá sekvence je tvořena 10 na sebe navazujícími akcemi od jednoho subjektu. Dataset sestává z následujících částí:

- RGB snímky ve formátu .jpg, s rozlišením 480×640
- Hloubkové mapy ve formátu .xml, rozlišení 320×240
- 3D záznamy kloubů ve formátu .txt
- Soubor s popisky, identifikující jednotlivé akce v zaznamenané sekvenci

V rámci této práce byly využity pouze záznamy kloubů v textovém formátu a dále soubor s popisky akcí. Záznamy kloubů jsou rozděleny do 20 textových souborů, každý soubor obsahuje všechny snímky z dané zaznamenané sekvence. Jeden řádek textu vždy představuje jeden snímek. První hodnota na řádku vždy udává číslo snímku, následující hodnoty pak popisují 3D pozice (x, y, z) jednotlivých kloubů. Pozice kloubů jsou udávány v metrech a reprezentují vzdálenost od snímače na zařízení.

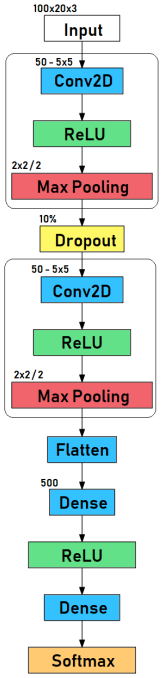
Druhým využitým datasetem je **MSR Action3D dataset** (MSR) [64]. Tento dataset je tvořen 20 akcemi: vysoké mávání rukou, horizontální mávání rukou, bití kladivem, chycení rukou, rána dopředu, hod, kreslení x, kreslení fajfky, kreslení kruhu, tleskání, mávání oběma rukama, boxování do strany, úklona, kop dopředu, kop do strany, běh, tenisový švih, tenisové podání, golfový švih a zvednutí a hod. Akce byly zaznamenány od celkem 10 subjektů. Každý subjekt zopakoval všechny akce dvakrát nebo třikrát. Celkově dataset obsahuje 567 zaznamenaných sekvencí. Dataset vytvořil Wanqing Li v době jeho působení v Microsoft Research Redmond a

byl zachycen zařízením s podobným senzorem jako má Microsoft Kinect. Data jsou v datasetu uložena ve třech formátech - hloubkové mapy, souřadnice kloubů relativní k displeji a souřadnice kloubů ve světových souřadnicích.

6.6 Modely konvolučních neuronových sítí

První implementovaný model vychází z architektury LeNet (schéma na obrázku 25). Vstupní vrstva této sítě přijímá vstupy s rozměry $100 \times 20 \times 3$. Následuje první konvoluční blok, který se skládá z konvoluční vrstvy následované aktivací ReLU a pooling vrstvou. Konvoluční vrstva má 50 filtrů s rozměry 5×5 a pooling vrstva má velikost okna 2×2 a krok 2. Tento blok je následován dropout vrstvou s mírou odpadu 10%. Následuje druhý konvoluční blok se stejnými parametry jako první blok. Síť je zakončena dvěmi plně propojenými vrstvami, vrstva flatten slouží pro zarovnání dat do 1D vektoru. První plně propojená vrstva je tvořena 500 neurony, velikost druhé pak závisí na počtu výstupů sítě (20 pro MSR a 10 pro UTKinect).

Rozměry vstupní vrstvy $100 \times 20 \times 3$ byly zvoleny kvůli charakteristice dat, se kterými se pracuje. Šířka a hloubka rozměrově odpovídají datům jednoho snímku v obou datasetech (ten je tvořen vždy 20×3 reálnými hodnotami reprezentující pozice kloubů). Výška pak byla zvolena tak, aby si jednotlivé vzorky mohly zachovat informace ze všech snímků a nedocházelo tak ke zbytečné ztrátě informací.



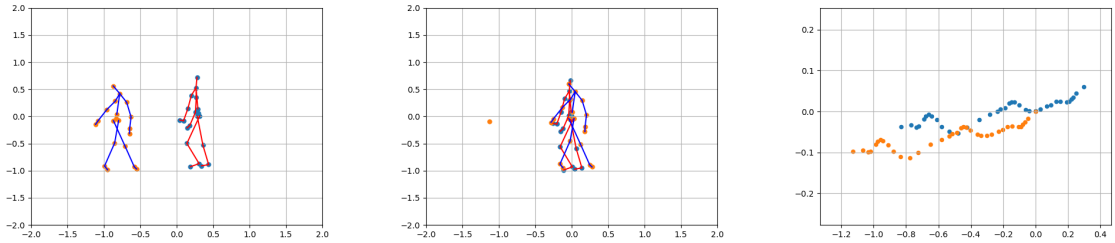
Obrázek 25: Schéma vlastního modelu sítě podle architektury LeNet.

Druhý model sítě vychází z architektury GoogLeNet. Tato síť byla sestavena podle původní publikace [34], některé parametry sítě však byly upraveny. Namísto optimizátoru SGD byl využit

optimizátor Adam. Dále byly změněny rozměry vstupních dat z $224 \times 224 \times 3$ na $128 \times 128 \times 3$. V první a třetí konvoluční vrstvě došlo ke snížení počtu filtrů z 64 na 48 (první) a ze 192 na 128 (třetí). Parametry incepčních modulů zůstaly zachovány. Ve vrstvě average pooling u nejhlubšího výstupu byly zmenšeny rozměry okna ze 7×7 na 4×4. Počty neuronů ve všech plně propojených vrstvách byly upraveny tak, aby odpovídaly jednotlivým datasetům. Tyto změny byly provedeny především kvůli snížení výpočetní náročnosti při trénování modelu. Schéma tohoto modelu odpovídá originálnímu modelu GoogLeNet (příloha B).

6.7 Předzpracování dat

Jelikož se u obou datasetů pracuje se stejným typem dat (pozice kloubů), byla pro oba datasety využita stejná metoda předzpracování dat. V první řadě byly přepočteny pozice kloubů v rámci jednotlivých akcí. V rámci datasetu byl zvolen jeden centrální kloub. Pozice všech ostatních kloubů v každé akci byly následně přepočteny relativně k pozici tohoto kloubu (obrázky 26a a 26b). V rámci další úpravy došlo u každé akce k odečtení pozice centrálního kloubu v prvním snímku od všech snímků v rámci akce. Tato úprava je ilustrována na obrázku 26c.



(a) Pozice kloubů před úpravami. (b) Pozice kloubů po úpravách. (c) Trajektorie centrálního kloubu.

Obrázek 26: Ilustrace úprav pozic kloubů akce *walk* z datasetu UTKinect. Obrázky a) a b) - červené spoje a modré body představují data prvního snímku akce. Modré spoje a orandžové body představují poslední snímek akce. V obrázku c) modré body představují pozice centrálního kloubu před úpravou.

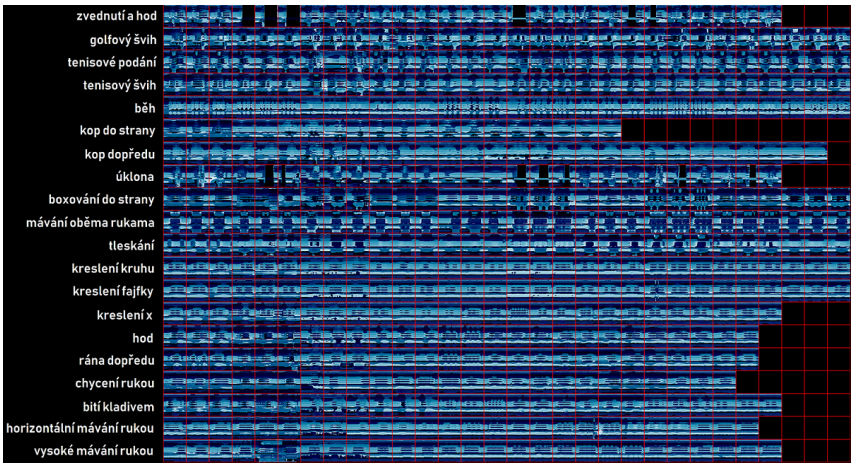
Pro možnost zpracování dat konvoluční sítí je třeba, aby měly jednotlivá vstupní data vždy stejné rozměry. K takové úpravě byla využita funkce *resize* z knihovny OpenCV, která slouží pro škálování pole do volitelných rozměrů. Jako interpolační metoda u této funkce byla využita bilineární interpolace.

Jelikož jsou snímky obou datasetů tvořeny trojrozměrnými záznamy pozic kloubů, byly jednotlivé snímky uspořádány do 2D polí s rozměry $klouby \times dim$, kde $klouby$ reprezentují celkový počet kloubů snímku (20 u obou datasetů) a dim reprezentuje trojrozměrnou souřadnici pozice kloubu. Spojením snímků v rámci akce pak vzniklo pole s rozměry $snimky \times klouby \times dim$, kde hodnota $snimky$ reprezentuje počet snímků pro danou akci. Na jednotlivé akce v tomto for-

mátu byla následně aplikována zmíněná funkce. Každá takto upravená akce pak měla rozměry $vyska \times sirka \times dim$, kde parametry $vyska$ a $sirka$ byly zadány jako argument funkce $resize$.

Z důvodu rozdílných velikostí vstupních vrstev obou sítí bylo třeba vygenerovat dvě rozdílné reprezentace akcí, obě s jinými parametry $vyska$ a $sirka$. Pro síť LeNet byly tyto parametry zvoleny $vyska = 100, sirka = 20$ a pro síť GoogLeNet $vyska = 128, sirka = 128$. Tyto parametry odpovídají rozměrům vstupních vrstev jednotlivých sítí.

Tyto operace pro předzpracování byly prováděny skriptem *convertDS.py*. Skript bylo nutné volat pouze jednou, výsledná předzpracovaná data byla následně uložena do složek *procDataset/* v adresářích datasetů. Při následném učení a testování sítí se pak pracovalo s již upravenými daty. Oba datasety byly v předzpracované formě uloženy ve formě 3 souborů - *actions.npy* a *actionsGooglenet.npy* obsahovaly samotné data akcí, *labels.npy* pak značky jednotlivých akcí. Na obrázku 27 je vyobrazen předzpracovaný dataset MSR. V obrázku jde vidět, že jednotlivé akce nemají v datasetu stejnou míru zastoupení. Například akce kop do strany se v datasetu vyskytuje pouze 20× zatímco akce běh se vyskytuje 30×. Černé oblasti u některých akcí jsou chybně zaznamenané části akcí.



Obrázek 27: Ilustrace již předzpracovaného datasetu MSR ve formě obrázku. Červená mřížka odděluje jednotlivé záznamy akcí, akce stejného typu jsou organizovány do řádků. Pro lepší viditelnost mřížky bylo využito barevné kódování $COLORMAP\_OCEAN$ z knihovny OpenCV. Větší verze je v příloze D.

6.8 Způsob testování

Po sestavení obou modelů byly sítě náležitě testovány. Pro účely testování byly použity rozdílné metody pro každý dataset. Pro urychlení trénování modelů byl využit školní server merlin1 s dvěma grafickýma kartama Nvidia GeForce RTX 2080.

K otestování sítí na MSR datasetu byla použita metoda zmíněná v [64]. Akce datasetu byly rozděleny do tří částí, každá část sestávala z 8 akcí. Akce byly rozděleny podle typu vykonávaných pohybů. AS1 a AS2 shlukují akce s podobnými pohyby. AS3 potom slučuje komplexní akce.

Detail rozdělení datasetu do částí je přiblížen v tabulce 2. Všechny skupiny akcí byly testovány třemi různými způsoby. Pro první test byla při trénování použita třetina vzorků, pro druhý test byly použity dvě třetiny vzorků a pro třetí byly použity vzorky pouze od poloviny subjektů. Jako dodatečné testování byly ještě všechny tři testy vykonány na všech akcích datasetu.

U datasetu UTKinect byl použit obdobný způsob testování, jako v originální publikaci datasetu [63]. Pro testování byla jednak použita metoda leave one sequence out cross validation (LOOCV). Tato metoda zahrnuje opakované trénování, kdy testovací část je vždy tvořena jen jedním vzorkem. Pro 200 vzorků je tedy nutné trénování opakovat 200×, pokaždé s rozdílným vzorkem. Dodatečně bylo provedeno i testování na celém datasetu najednou s poměrem trénovacích a testovacích dat 1 : 1. U této metody byly akce rozděleny podle jednotlivých subjektů - každá z částí pracovala se všemi akcemi od 5 subjektů. Tento způsob testování je oproti LOOCV více vyzývajícím, jelikož má síť k dispozici menší množství dat pro trénování.

Testování obou datasetů i sítí bylo zopakováno 20× a výsledky byly zprůměrovány. Síť LeNet byla trénována pouze při 15 iteracích na obou datasetech. U sítě GoogLeNet bylo zjištěno, že pro porovnávatelné výsledky vyžaduje daleko vyšší počet iterací než síť LeNet. Z tohoto důvodu bylo zvoleno 120 iterací na datasetu MSR a 80 iterací na datasetu UTKinect. Tyto hodnoty byly odvozeny experimentováním.

Set akcí 1 (AS1)	Set akcí 2 (AS2)	Set akcí 3 (AS3)
Horizontální mávnutí rukou	Vysoké mávnutí rukou	Hod
Bití kladivem	Chycení rukou	Kop dopředu
Rána dopředu	Kreslení x	Kop do strany
Hod	Kreslení fajfky	Běh
Tleskání	Kreslení kruhu	Tenisový švih
Úklona	Mávání oběma rukama	Tenisové podání
Tenisové podání	Kop dopředu	Golfový švih
Zvednutní a hod	Boxování do stany	Zvednutí a hod

Tabulka 2: Tabulka ukazující jednotlivé části datasetu MSR použíté při testování. [64].

6.9 Výsledky a porovnání

Trénování i testování obou sítí proběhlo bez obtíží. Výsledné hodnoty z testování sítí na datasetu MSR jsou v tabulce 3. V tabulce je vidět, že obecné vyšší přesnosti dosáhla síť GoogLeNet. Tento výsledek se kvůli komplexitě a hloubce sítě GoogLeNet (v porovnání s LeNet) dal očekávat. Síť dosáhla výrazně lepších výsledku především při testování s využitím dvou třetin vzorků pro trénování a dále také v jednotlivých testech na všech akcích najednou.

	Test1		Test 2		Test 3	
	LeNet	GoogLeNet	LeNet	GoogLeNet	LeNet	GoogLeNet
AS1	79.5%	**85.8%**	81.5%	**93.1%**	**67.3%**	66.2%
AS2	**79.5%**	76.6%	88.4%	**92.7%**	**72.0%**	71.9%
AS3	**84.7%**	78.5%	89.3%	**95.3%**	74.4%	**78.2%**
Průměr	**81.2%**	80.3%	86.4%	**93.7%**	71.2%	**72.1%**
Všechny akce	74.5%	**87.7%**	79.0%	**93.1%**	60.2%	**61.4%**

Tabulka 3: Tabulka přesnosti obou sítí na datasetu MSR.

Výsledky na datasetu UTKinect jsou zachyceny v tabulce 4 a na obrázcích 28. Obrázky představují konfúzní matice jednotlivých sítí pro oba druhy validace. Zobrazené konfúzní matice byly vytvořeny zprůměrováním konfúzních matic ze všech 20 trénovacích iterací. V tabulce je pak zapsáno F_1 skóre (harmonický průměr) pro každou akci individuálně. Pro výpočet F_1 skóre byl použit následující vztah:

$$F_1 = \frac{2TP}{2TP + FP + FN}, \tag{11}$$

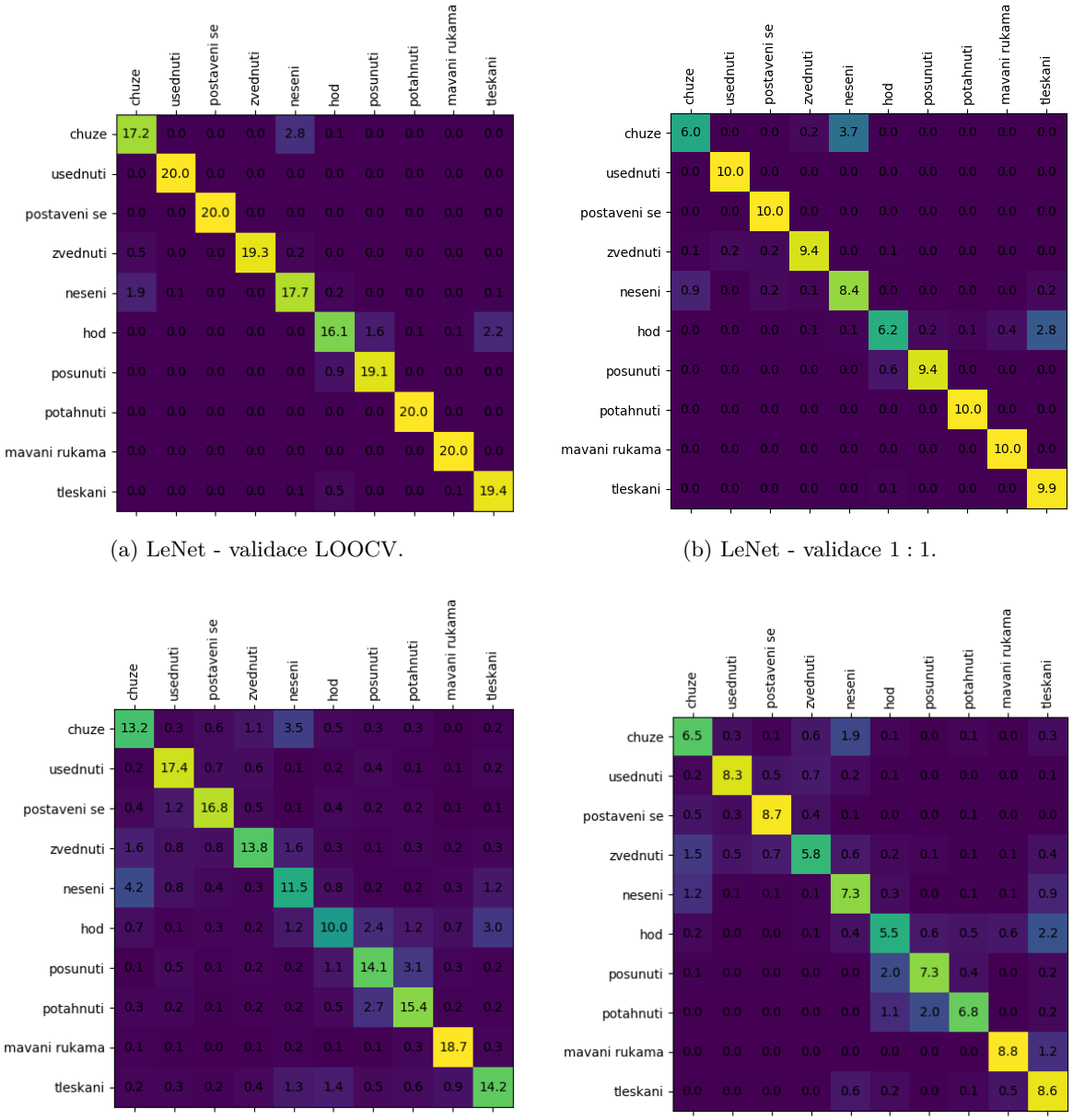
kde TP je počet skutečně pozitivních hodnot (true positive), FP je počet falešně pozitivních hodnot (false positive) a FN je počet falešně negativních hodnot (false negative). Tyto hodnoty mohou být rovněž vypozorovány z konfúzní matice v sloupci a řádku dané akce. [65]

Akce	LOOCV		Validace 1:1	
	LeNet	GoogLeNet	LeNet	GoogLeNet
Chůze	87.1%	64.4%	70.8%	64.5%
Sednutí	99.9%	82.9%	98.8%	85.1%
Postavení se	100%	84.0%	97.8%	86.8%
Zvednutí předmětu	98.2%	73.6%	94.7%	66.3%
Nesení předmětu	86.9%	57.6%	75.7%	69.4%
Hod	85.1%	56.7%	73.5%	56.3%
Strčení	93.8%	68.5%	95.7%	72.8%
Potáhnutí	99.9%	73.6%	99.3%	74.8%
Mávání rukama	99.6%	90.1%	98.0%	88.2%
Zatleskání	92.9%	70.8%	86.3%	71.3%
Celkově	94.4%	72.5%	89.4%	73.5%

Tabulka 4: Tabulka F_1 skóre pro každou akci datasetu UTKinect.

V rámci testů na tomto datasetu dosáhla výrazně vyšší úspěšnosti síť LeNet. Tento fakt je

pravděpodobně způsoben nedostatečným počtem iterací při trénování sítě GoogLenet. Nejlépe odhadovanými akcemi jsou mávání rukama a postavení se. Nejproblémovější akce jsou u obou sítí hod, nesení předmětu a chůze. Akce hod je často zaměňována za akci tleskání. Akce chůze a nesení předmětu jsou pak navzájem zaměňovány. To je pravděpodobně způsobeno tím, že obě akce jsou si dosti podobné, jelikož při nesení také dochází k chůzi.



(a) LeNet - validace LOOCV.

(b) LeNet - validace 1 : 1.

(c) GoogLeNet - validace LOOCV.

(d) GoogLeNet - validace 1 : 1.

Obrázek 28: Výsledné průměrné konfúzní matice obou sítí.

Rozložení výsledků je pro oba typy validací podobné. Síť LeNet dosáhla nižší úspěšnosti při validaci 1:1 (především u zmíněných problémových akcí). Síť GoogLeNet pak dosáhla mírně nižší

úspěšnosti při validaci LOOCV.

Z hlediska doby trénování i evaluace je síť LeNet bezesporu efektivnější než GoogLeNet. Tyto údaje jsou zobrazeny v tabulkách 5 a 6. Časové údaje byly zaznamenávány na školním serveru merlin1. Z tabulek je patrné, že síť LeNet je výrazně rychlejší hlavně při trénování na obou datasetech. Tento jev je zapříčiněn tím, že síť GoogLeNet je výrazně hlubší. Rozdíl časů u trénování roste s obtížností jednotlivých typů testování. Nejvíce času vužaduje validace LOOCV u dataset UTKinect. U evaluace pak rozdíl není až tak zásadní, stále má však síť LeNet mírně navrch.

Sítě	MSR		UTKinect	
	Sety akcí	Všechny akce	LOOCV	Validace 1:1
LeNet	$8.46s$	$7.02s$	$190.03s$	$4.34s$
GoogLeNet	$210.46s$	$143.59s$	$2h$	$42.5s$

Tabulka 5: Tabulka přibližných časů trénovaní sítí.

Sítě	MSR	UTKinect
LeNet	$2.24s$	$2.16s$
GoogLeNet	$3.57s$	$3.59s$

Tabulka 6: Tabulka přibližných časů predikce sítí.

7 Závěr

Cílem této práce bylo seznámení s metodami Deep Learningu aplikované na rozpoznávání lidských činností. V prvních kapitolách byly popsány principy neuronových sítí. Byla zde popsána biologická inspirace za těmito systémy, základní strutura, způsoby učení i některé konkrétní typy neuronových sítí. V dalších kapitolách pak byly blíže popsány konvoluční neuronové sítě. U těchto sítí byly popsány především typy vrstev a také některé významné architektury těchto sítí. V další kapitole potom byly popsány principy detekce a klasifikace akcí včetně popisu několika konkrétních metod jako Histogram orientovaných gradientů (HOG) nebo Časoprostorové zájmové body (STIP).

Poslední kapitola potom byla věnována implementaci a testování vlastních modelů konvolučních neuronových sítí. Pro tyto účely byly vytvořeny dva modely konvolučních neuronových sítí podle architektur LeNet a GoogLeNet. Tyto modely byly následně testovány na datasetech lidských akcí MSR Action3D dataset a UTKinect-Action3D dataset.

Ve výsledku byla u datasetu MSR úspěšnější síť GoogLeNet, zatímco u datasetu UTKinect dosáhla lepších výsledků síť LeNet. Z těchto výsledků jsem došel k zjištění, že výsledná přesnost trénované sítě nezáleži nutně jen na hloubce a komplexitě dané sítě. Je nutno vzít v potaz mnohem více parametrů jako počet iterací při tréninku nebo i vlastnosti datasetu, jako například celkové množství dat. Méně komplexní síť pak může být vhodnější volbou zejména v případech, kdy záleží na době potřebné pro trénování sítě.

Vývoj této bakalářské práce mi pomohl zdokonalit mé znalosti jazyka Python a uvedl mě do problematiky neuronových sítí a rozpoznávání akcí. Zároveň jsem se obeznámil s frameworkem Tensorflow.

Literatura

[1] SHIFFMAN, Daniel. Chapter 10. Neural networks. *The Nature of Code* [online]. 2012 [cit. 2019-04-07]. ISBN 978-0985930806. Dostupné z: http://natureofcode.com/book/chapter-10-neural-networks/

[2] SAYAD, Saed. Artificial Neural Network. *An Introduction to Data Science* [online]. c2010-2019 [cit. 2019-04-07]. Dostupné z: https://www.saedsayad.com/artificial_neural_network.htm

[3] VAN VEEN, Fjodor. Neural Network Zoo Prequel: Cells and layers. In: *The Asimov Institute* [online]. Europalaan 20, 3526 KS Utrecht The Netherlands, c2019, 2017-03-31 [cit. 2019-04-07]. Dostupné z: http://www.asimovinstitute.org/neural-network-zoo-prequel-cells-layers/

[4] VAN VEEN, Fjodor. The Neural Network Zoo. In: *The Asimov Institute* [online]. Europalaan 20, 3526 KS Utrecht The Netherlands, c2019, 2016-09-14 [cit. 2019-04-07]. Dostupné z: http://www.asimovinstitute.org/neural-network-zoo/

[5] ANDERSON, Dave a George MCNEILL. *Artificial Neural Network Technology* [online]. 1992-09-20 [cit. 2019-04-08]. Dostupné z: http://www.psych.utoronto.ca/users/reingold/courses/ai/cache/neural2.html

[6] Overview of neuron structure and function. In: *Khan Academy* [online]. Mountain View, California: Khan Academy, c2019 [cit. 2019-04-08]. Dostupné z: https://www.khanacademy.org/science/biology/human-biology/neuron-nervous-system/a/overview-of-neuron-structure-and-function

[7] CHALUPNÍK, Vitalij. Biologické algoritmy (4) - Neuronové sítě. In: *Root.cz* [online]. 2012-04-25 [cit. 2019-04-07]. Dostupné z: https://www.root.cz/clanky/biologicke-algoritmy-4-neuronove-site/

[8] FUMO, David. A Gentle Introduction To Neural Networks Series—Part 1. In: *Towards Data Science* [online]. 2017-08-04 [cit. 2019-04-07]. Dostupné z: https://towardsdatascience.com/a-gentle-introduction-to-neural-networks-series-part-1-2b90b87795bc

[9] KAWAGUCHI, Kiyoshi. *A multithread software model for backpropagation neural network applications* [online]. El Paso, Texas, U.S., 2000 [cit. 2019-04-07]. Dostupné z: http://wwwold.ece.utep.edu/research/webfuzzy/docs/kk-thesis/kk-thesis-html/thesis.html. Diplomová práce. The University of Texas at El Paso.

[10] A Quick Introduction to Neural Networks. In: *The data science blog* [online]. Ujjwal Karn, August 9, 2016 [cit. 2019-04-07]. Dostupné z: https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/

[11] KARÁSEK, Štěpán. *Neuronové sítě a genetické algoritmy* [online]. Brno, 2016 [cit. 2019-04-27]. Dostupné z: https://core.ac.uk/download/pdf/44404703.pdf. Diplomová práce. Vysoké učení technické v Brně.

[12] KURAMA, Vihar. Introduction To Machine Learning. In: *Towards Data Science* [online]. Jul 15, 2017 [cit. 2019-04-07]. Dostupné z: https://towardsdatascience.com/introduction-to-machine-learning-db7c668822c4

[13] A brief introduction to reinforcement learning. In: *FreeCodeCamp* [online]. San Francisco, California: Free Code Camp, Aug 27, 2018 [cit. 2019-04-07]. Dostupné z: https://medium.freecodecamp.org/a-brief-introduction-to-reinforcement-learning-7799af5840db

[14] *Backpropagation* [online]. Anthony Papagelis [cit. 2019-04-07]. Dostupné z: https://www.cse.unsw.edu.au/ cs9417ml/MLP2/BackPropagation.html

[15] HLOŽEK, Bohuslav. *Bayesovské a neuronové sítě* [online]. Brno, 2017 [cit. 2019-04-07]. Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=158331. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií.

[16] Overfitting. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-04-07]. Dostupné z: https://en.wikipedia.org/wiki/Overfitting

[17] NELSON, Daniel. Neural Net Dropout: Dealing With Overfitting. In: *DataScience* [online]. c2018, Jan 21, 2018 [cit. 2019-04-27]. Dostupné z: https://www.datascience.us/neural-net-dropout-dealing-overfitting/

[18] 7. The backpropagation algorithm. ROJAS, Raul. *Neural Networks - A Systematic Introduction* [online]. Berlin, New-York: Springer-Verlag, 1996, s. 151-153 [cit. 2019-04-27]. ISBN 978-3-642-61068-4. Dostupné z: https://page.mi.fu-berlin.de/rojas/neural/

[19] CHAPTER 2: How the backpropagation algorithm works. NIELSEN, Michael. *Neural Networks and Deep Learning* [online]. Determination Press, 2015 [cit. 2019-04-27]. Dostupné z: http://neuralnetworksanddeeplearning.com/

[20] Neuronové sítě. In: *CourseWare* [online]. Praha: ČVUT [cit. 2019-04-27]. Dostupné z: https://cw.fel.cvut.cz/old/_media/courses/a6m33dvz/dvz2017-05-nnet.pdf

[21] Chapter 9: Convolutional Networks. GOODFELLOW, Ian, Yoshua BENGIO a Aaron COURVILLE. *Deep Learning* [online]. MIT Press, 2016, s. 326-333 [cit. 2019-04-28]. Dostupné z: http://www.deeplearningbook.org

[22] BHANDARE, Ashwin, Maithili BHIDE, Pranav GOKHALE a Rohan CHANDAVARKAR. Applications of Convolutional Neural Networks. *International Journal of Computer Science*

and Information Technologies [online]. **2016**, 2206-2215 [cit. 2019-04-28]. ISSN 0975-9646. Dostupné z: http://ijcsit.com/docs/Volume%207/vol7issue5/ijcsit20160705014.pdf

[23] HUBEL, D. a T. WIESEL. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology* [online]. 1962, **1962**(160), 106-154 [cit. 2019-04-09]. 14449617. Dostupné z: https://physoc.onlinelibrary.wiley.com/doi/pdf/10.1113/jphysiol.1962.sp006837

[24] FERGUS, Rob. *Neural Networks* [online]. In: . Machine Learning Summer Schools, 2015 [cit. 2019-04-28]. Dostupné z: http://mlss.tuebingen.mpg.de/2015/slides/fergus/Fergus_1.pdf

[25] LECUN, Y., L. BOTTOU, Y. BENGIO a P. HAFFNER. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* [online]. 1998, **86**(11), 2278-2324 [cit. 2019-04-28]. DOI: 10.1109/5.726791. ISSN 00189219. Dostupné z: http://ieeexplore.ieee.org/document/726791/

[26] KIMBALL, John W. Processing Visual Information. In: *Kimball's Biology Pages* [online]. 20 May 2011 [cit. 2019-04-28]. Dostupné z: http://www.biology-pages.info/V/VisualProcessing.html

[27] FUKUSHIMA, Kunihiko. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics* [online]. 1980, **36**(4), 193-202 [cit. 2019-04-28]. DOI: 10.1007/BF00344251. ISSN 0340-1200. Dostupné z: http://link.springer.com/10.1007/BF00344251

[28] Convolutional Neural Network. In: *Deep Learning: Computer Science Department, Stanford University* [online]. [cit. 2019-04-28]. Dostupné z: http://ufldl.stanford.edu/tutorial/supervised/ConvolutionalNeuralNetwork/

[29] LIU, Jeremy, Federico SPEDALIERI, Ke-Thia YAO, et al. Adiabatic Quantum Computation Applied to Deep Learning Networks. *Entropy* [online]. 2018, **20**(5) [cit. 2019-04-28]. DOI: 10.3390/e20050380. ISSN 1099-4300. Dostupné z: http://www.mdpi.com/1099-4300/20/5/380

[30] YU, Dingjun, Hanli WANG, Peiqiu CHEN a Zhihua WEI. Mixed Pooling for Convolutional Neural Networks. *Rough Sets and Knowledge Technology* [online]. Cham: Springer International Publishing, 2014, 2014, , 364-375 [cit. 2019-04-28]. Lecture Notes in Computer Science. DOI: 10.1007/978-3-319-11740-9_34. ISBN 978-3-319-11739-3. Dostupné z: http://link.springer.com/10.1007/978-3-319-11740-9_34

[31] KRIZHEVSKY, Alex, Ilya SUTSKEVER a Geoffrey E. HINTON. ImageNet classification with deep convolutional neural networks. *Communications of the ACM* [online]. 2017, **60**(6), 84-90 [cit. 2019-04-28]. DOI: 10.1145/3065386. ISSN 00010782. Dostupné z: http://dl.acm.org/citation.cfm?doid=3098997.3065386

[32] ALOM, Md Zahangir. *The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches* [online]. , 39 [cit. 2019-04-28]. Dostupné z: https://arxiv.org/abs/1803.01164

[33] TSANG, Sik-Ho. Review: AlexNet, CaffeNet—Winner of ILSVRC 2012. In: *Medium* [online]. Aug 9, 2018 [cit. 2019-04-28]. Dostupné z: https://medium.com/coinmonks/paper-review-of-alexnet-caffenet-winner-in-ilsvrc-2012-image-classification-b93598314160

[34] SZEGEDY, Christian, WEI LIU, YANGQING JIA, et al. Going deeper with convolutions. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* [online]. IEEE, 2015, 2015, , 1-9 [cit. 2019-04-28]. DOI: 10.1109/CVPR.2015.7298594. ISBN 978-1-4673-6964-0. Dostupné z: http://ieeexplore.ieee.org/document/7298594/

[35] KARPATHY, Andrej. *Convolutional Neural Networks* [online]. In: . [cit. 2019-04-28]. Dostupné z: http://cs231n.github.io/convolutional-networks/

[36] BHARATH, Raj. A Simple Guide to the Versions of the Inception Network. In: *Towards Data Science* [online]. May 29, 2018 [cit. 2019-04-28]. Dostupné z: https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202

[37] HE, Kaiming, Xiangyu ZHANG, Shaoqing REN a Jian SUN. Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* [online]. IEEE, 2016, 2016, , 770-778 [cit. 2019-04-28]. DOI: 10.1109/CVPR.2016.90. ISBN 978-1-4673-8851-1. Dostupné z: http://ieeexplore.ieee.org/document/7780459/

[38] GROSS, Sam a Michael WILBER. Training and investigating Residual Nets. In: *Torch* [online]. February 4, 2016 [cit. 2019-04-28]. Dostupné z: http://torch.ch/blog/2016/02/04/resnets.html

[39] KONG, Yu a Yun FU. *Human Action Recognition and Prediction: A Survey* [online]. 28 Jun 2018, , 20 [cit. 2019-04-28]. Dostupné z: https://arxiv.org/abs/1806.11230

[40] DAS DAWN, Debapratim a Soharab Hossain SHAIKH. A comprehensive survey of human action recognition with spatio-temporal interest point (STIP) detector. *The Visual Computer* [online]. 2016, **32**(3), 289-306 [cit. 2019-04-28]. DOI: 10.1007/s00371-015-1066-2. ISSN 0178-2789. Dostupné z: http://link.springer.com/10.1007/s00371-015-1066-2

[41] POPPE, Ronald. A survey on vision-based human action recognition. *Image and Vision Computing* [online]. 2010, **28**(6), 976-990 [cit. 2019-04-28]. DOI: 10.1016/j.imavis.2009.11.014. ISSN 02628856. Dostupné z: https://linkinghub.elsevier.com/retrieve/pii/S0262885609002704

[42] KIM, Tae Soo a Austin REITER. Interpretable 3D Human Action Analysis with Temporal Convolutional Networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* [online]. IEEE, 2017, 2017, , 1623-1631 [cit. 2019-04-28]. DOI: 10.1109/CVPRW.2017.207. ISBN 978-1-5386-0733-6. Dostupné z: http://ieeexplore.ieee.org/document/8014941/

[43] QI, Jin, Zhiyong YANG a Marco CRISTANI. Learning Dictionaries of Sparse Codes of 3D Movements of Body Joints for Real-Time Human Activity Understanding. *PLoS ONE* [online]. 2014, **9**(12), 24 [cit. 2019-04-28]. DOI: 10.1371/journal.pone.0114147. ISSN 1932-6203. Dostupné z: https://dx.plos.org/10.1371/journal.pone.0114147

[44] DU, Yong, Yun FU a Liang WANG. Skeleton based action recognition with convolutional neural network. *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)* [online]. IEEE, 2015, 2015, , 579-583 [cit. 2019-04-28]. DOI: 10.1109/ACPR.2015.7486569. ISBN 978-1-4799-6100-9. Dostupné z: http://ieeexplore.ieee.org/document/7486569/

[45] LAPTEV, Ivan. On Space-Time Interest Points. *International Journal of Computer Vision* [online]. 2005, **64**(2-3), 107-123 [cit. 2019-04-28]. DOI: 10.1007/s11263-005-1838-7. ISSN 0920-5691. Dostupné z: http://link.springer.com/10.1007/s11263-005-1838-7

[46] SEIDENARI, Lorenzo. *Spatio temporal features* [online]. In: . University of Florence, 2010 [cit. 2019-04-28]. Dostupné z: http://www.micc.unifi.it/seidenari/wp-content/uploads/2010/01/A51-Spatio-temporal-features1.pdf

[47] SIMAC-LEJEUNE, Alain. *Moving object analysis in video sequences using space-time interest points* [online]. , 201-204 [cit. 2019-04-28]. Dostupné z: https://hal.archives-ouvertes.fr/hal-00674642/document

[48] DALAL, N. a B. TRIGGS. Histograms of Oriented Gradients for Human Detection. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)* [online]. IEEE, 2005, , 886-893 [cit. 2019-04-28]. DOI: 10.1109/CVPR.2005.177. ISBN 0-7695-2372-2. Dostupné z: http://ieeexplore.ieee.org/document/1467360/

[49] CRNOJEVIĆ, Vladimir, Marko PANIĆ, Branko BRKLJAČ, Dubravko ĆULIBRK, Jelena AČANSKI a Ante VUJIĆ. Image Processing Method for Automatic Discrimination of Hoverfly Species. *Mathematical Problems in Engineering* [online]. 2014, **2014**, 1-12 [cit. 2019-04-28]. DOI: 10.1155/2014/986271. ISSN 1024-123X. Dostupné z: http://www.hindawi.com/journals/mpe/2014/986271/

[50] JUNEJO, I N, E DEXTER, I LAPTEV a Patrick PÉREZ. View-Independent Action Recognition from Temporal Self-Similarities. *IEEE Transactions on Pattern Analysis and Machine Intelligence* [online]. 2011, **33**(1), 172-185

[cit. 2019-04-28]. DOI: 10.1109/TPAMI.2010.68. ISSN 0162-8828. Dostupné z: http://ieeexplore.ieee.org/document/5432213/

[51] PADMANABHA, Akshay a Christopher WILLIAMS. K-nearest Neighbors. In: *Brilliant* [online]. c2019 [cit. 2019-04-28]. Dostupné z: https://brilliant.org/wiki/k-nearest-neighbors/

[52] AISEN, Ben. *A Comparison of Multiclass SVM Methods* [online]. December 15, 2006 [cit. 2019-04-28]. Dostupné z: https://courses.media.mit.edu/2006fall/mas622j/Projects/aisen-project/

[53] MANOSHA CHATHURAMALI, K. G. a Ranga RODRIGO. Faster human activity recognition with SVM. *International Conference on Advances in ICT for Emerging Regions (ICTer2012)* [online]. IEEE, 2012, 2012, , 197-203 [cit. 2019-04-28]. DOI: 10.1109/ICTer.2012.6421415. ISBN 978-1-4673-5530-8. Dostupné z: http://ieeexplore.ieee.org/document/6421415/

[54] DONAHUE, Jeff, Lisa Anne HENDRICKS, Marcus ROHRBACH, Subhashini VENUGOPALAN, Sergio GUADARRAMA, Kate SAENKO a Trevor DARRELL. Long-Term Recurrent Convolutional Networks for Visual Recognition and Description. *IEEE Transactions on Pattern Analysis and Machine Intelligence* [online]. 2017, **39**(4), 677-691 [cit. 2019-04-28]. DOI: 10.1109/TPAMI.2016.2599174. ISSN 0162-8828. Dostupné z: http://ieeexplore.ieee.org/document/7558228/

[55] LEA, Colin, Michael D. FLYNN, Rene VIDAL, Austin REITER a Gregory D. HAGER. Temporal Convolutional Networks for Action Segmentation and Detection. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* [online]. IEEE, 2017, 2017, , 1003-1012 [cit. 2019-04-29]. DOI: 10.1109/CVPR.2017.113. ISBN 978-1-5386-0457-1. Dostupné z: http://ieeexplore.ieee.org/document/8099596/

[56] CHOLLET, François. Keras. *Keras* [online]. .: François Chollet, 2015 [cit. 2019-03-30]. Dostupné z: https://keras.io/

[57] Tensorflow. *Tensorflow: Large-Scale Machine Learning on Heterogeneous Systems* [online]. Martín Abadi, 2015 [cit. 2019-03-30]. Dostupné z: https://tensorflow.org

[58] Google TPU. In: *Google Cloud Platform* [online]. Mountain View: Google, 2016 [cit. 2019-03-19]. Dostupné z: https://cloud.google.com/blog/products/gcp/google-supercharges-machine-learning-tasks-with-custom-chip

[59] Tensorflow Architecture. In: *Tensorflow* [online]. [cit. 2019-04-29]. Dostupné z: https://www.tensorflow.org/guide/extend/architecture

[60] OpenCV Library. *OpenCV* [online]. Intel Corporation, 2000 [cit. 2019-03-30]. Dostupné z: https://opencv.org/

[61] *Nvidia CUDA Home Page* [online]. Nvidia Corporation [cit. 2019-03-31]. Dostupné z: https://developer.nvidia.com/cuda-zone

[62] ABI-CHAHLA, Fedy. Nvidia's CUDA: The End of the CPU?. *Tom's Hardware* [online]. 2008 [cit. 2019-03-31]. Dostupné z: https://www.tomshardware.com/reviews/nvidia-cuda-gpu,1954.html

[63] XIA, Lu, Chia-Chih CHEN a J. K. AGGARWAL. View invariant human action recognition using histograms of 3D joints. *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops* [online]. IEEE, 2012, 2012, , 20-27 [cit. 2019-04-29]. DOI: 10.1109/CVPRW.2012.6239233. ISBN 978-1-4673-1612-5. Dostupné z: http://ieeexplore.ieee.org/document/6239233/

[64] LI, Wanqing, Zhengyou ZHANG a Zicheng LIU. Action recognition based on a bag of 3D points. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops* [online]. IEEE, 2010, 2010, , 9-14 [cit. 2019-04-29]. DOI: 10.1109/CVPRW.2010.5543273. ISBN 978-1-4244-7029-7. Dostupné z: http://ieeexplore.ieee.org/document/5543273/

[65] Confusion matrix. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-04-29]. Dostupné z: https://en.wikipedia.org/wiki/Confusion_matrix

A Dokumentace kódu

Projekt je pro přehlednost organizován do několika složek. V kořenovém adresáři se nachází několik základních skriptů, které slouží pro ovládání celé aplikace:

- **convertDS.py** je určen pro generování předzpracovaných verzí datasetů.
- **train.py** slouží pro trénování, testování a ukládání výsledků sítí.
- **printResults.py** zobrazuje zaznamenané výsledky z trénování.
- **evaluate.py** je určen pro využívání již vytrénovaného modelu sítě pro predikci.

Všechny tyto skripty podporují argumenty $-h/--help$, které vypíšou bližší info o možných parametrech skriptů. V ukázce 1 jsou vypsány základní tvary těchto skriptů.

```
#spusti generovani predzpracovanych verzi obou datasetu
$ python convertDS.py --both

#zahaji trenovani site LeNet na datasetu MSR; pro vypocty bude pouzita GPU0
$ python train.py -m lenet -d MSR --gpu 0

#vypise data z testovani site googlenet na ds UTC; metoda validace je 1:1
#rovnez vykresli konfuzni matici (vyzaduje matplotlib)
$ python printResults.py -m googlenet -d UTC --alt --plt

#vyhodnoti predikci pro data ze souboru data.bin
$ python evaluate.py -d UTC -m lenet -w lenetWeights.h5 -if data.bin
```

Výpis 1: Příklady volání centrálních skriptů projektu.

Všechen ostatní obsah projektu je organizován do tří složek:

- složky **UTKinect/** a **MSR/** obsahují skripty a data vztažené k jednotlivým datasetům.
- složka **models/** obsahuje definice modelů obou sítí.

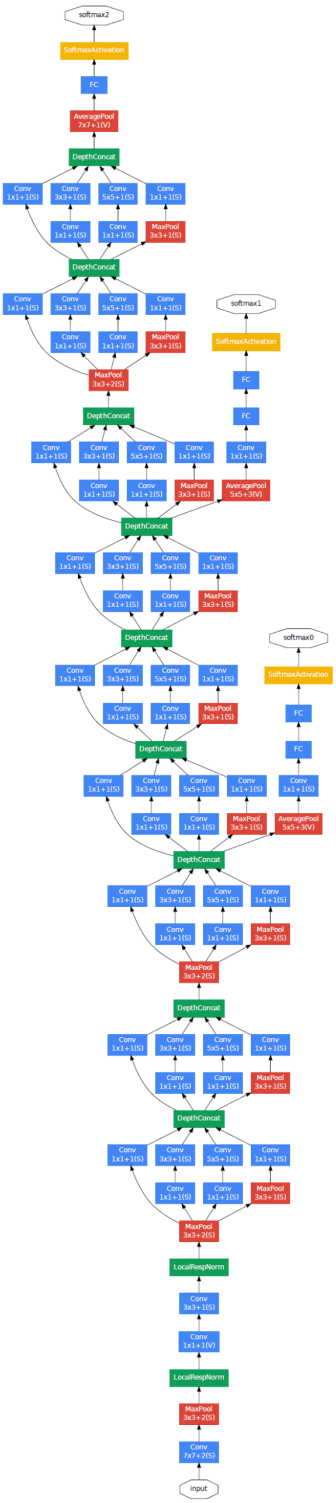
Se skripty a zdroji obsaženými v těchto složkách dále pracují centrální skripty.

V rámci obou datasetových složek se dále nacházejí složky *dataset/*, *procDataset/* a *utility/*. Složka *utility* obsahuje dodatečné skripty, které byly využity pro generování obsahu (obrázků) pro tuto bakalářskou práci. Do složky *procDataset/* se generuje předzpracovaná verze datasetu. Ve složce *dataset/* se pak nachází základní verze datasetu (nebo alespoň jeho důležité části).

Pro vytvoření předzpracovaných verzí se musí ve složce *dataset/* nacházet složka *joints/* s jednotlivými soubory s popisy kloubů. U datasetu UTKinect se ve složce dále musí nacházet

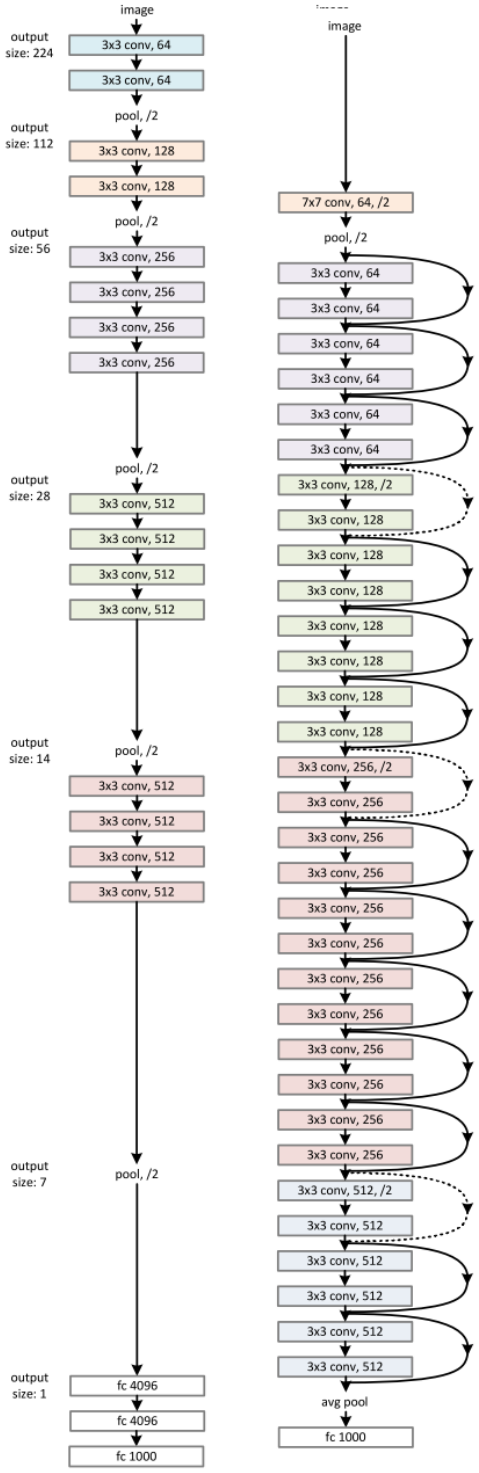
soubor *actionLabel.txt* s popisy značek akcí. U datasetu MSR se ve složce musí nacházet soubor *JiangExperimentFileList.txt*, ve kterém jsou vypsány názvy využitých souborů z datasetu.

B GoogleNet



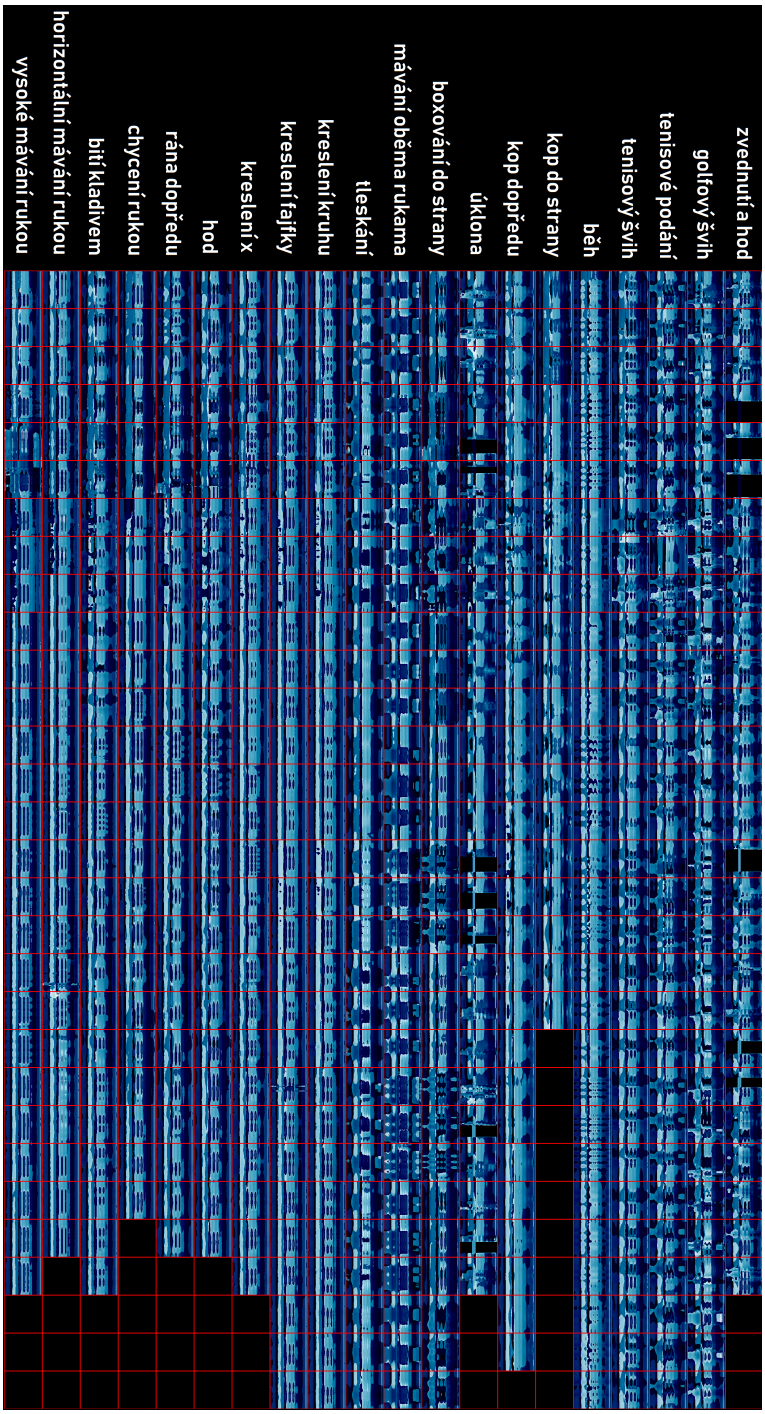
Obrázek 29: Architektura sítě GoogleNet. Převzato z [34].

C ResNet



Obrázek 30: Porovnání architektur VGG-19 (v levo) a ResNet (v pravo). Převzato z [37].

D Dataset MSR ve formě obrázku



Obrázek 31: Ilustrace předzpracovaného datasetu MSR ve formě obrázku. Červená mřížka odděluje jednotlivé záznamy akcí, akce stejného typu jsou organizovány do sloupců. Černé úseky jsou chybné části akcí nebo dorovnání do počtu akcí. Pro lepší viditelnost mřížky bylo využito barevné kódování $COLORMAP\_OCEAN$ z knihovny OpenCV.